Juan José Pardo Mateo

DESCRIPCIÓN E IMPLEMENTACIÓN DE TPPAL: UN ÁLGEBRA DE PROCESOS TEMPORIZADOS Y PROBABILÍSTICOS

I.S.B.N. Ediciones de la UCLM
84-8427-459-4

Ediciones de la Universidad
de Castilla-La Mancha

Cuenca, 2006

UNIVERSIDAD DE CASTILLA-LA MANCHA

Departamento de Informática

DESCRIPCIÓN E IMPLEMENTACIÓN DE TPPAL: UN ÁLGEBRA DE PROCESOS TEMPORIZADOS Y PROBABILÍSTICOS

Tesis Doctoral

Presentada por:

Juan José Pardo Mateo

Dirigida por

Valentín Valero Ruiz

Fernando Cuartero Gómez

Albacete: Julio de 2003

DESCRIPCIÓN E IMPLEMENTACIÓN DE TPPAL: UN ÁLGEBRA DE PROCESOS TEMPORIZADOS Y PROBABILÍSTICOS

Juan José Pardo Mateo

Julio 2003

Tesis Doctoral presentada al Departamento de Informática de la Universidad de Castilla-La Mancha para la obtención del grado de Doctor en Informática

Agradecimientos

Antes de iniciar esta tesis, deseo mostrar mi más sincero agradecimiento a las siguientes personas sin las cuales no habría conseguido terminar este proyecto en el que me embarqué hace unos años:

... A mis directores de tesis, Valentín y Fernando, por el consejo y ayuda que me han prestado durante mis estudios de doctorado, por el tiempo que les he hecho perder y por la paciencia que han tenido conmigo durante la elaboración de esta tesis. Gracias a los dos.

... a mis padres Lorenzo e Isabel, a mi hermana Isabel Mari y a mi hermano Lorenzo que han estado a mi lado apoyandome en todo momento.

... a Diego y Mari Carmen por los animos que tantas veces me han dado, y por su disposición a ayudarme en todo lo que he necesitado.

... a mis compañeros del Departamentos de Informática, en especial a los de las "mazmorras", con los cuales en los últimos seis años he compartido clases, cafés y alguna que otra cena " baja en calorías y colesterol".

... a Dios, que me ha dado fuerza y salud para llevar adelante mi trabajo.

... a Paco Pepe, Martín, Diego, Javier Campos y Javier Oliver por aceptar ser miembros del tribunal y por haber perdido parte de su tiempo leyendo este conjunto de páginas no excesivamente ameno.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

1.1. Modelado de sistemas concurrentes

Quedan ya lejanos los tiempos en los que la programación de los computadores se hacía manualmente y con elevados tiempos de ejecución para pequeñas rutinas de cálculo. Hoy nos encontramos incluso con máquinas dotadas de miles de procesadores capaces de realizar en unos segundos cálculos mucho más complejos, como inversión de grandes matrices, procesamiento de imágenes, etc; y sistemas de tiempo real, diseñados para responder a ciertos eventos externos en un plazo de tiempo muy limitado. En todos estos ejemplos aparecen de forma natural varios procesos concurrentes que cooperan entre sí para la consecución de su objetivo final, que es conseguir el funcionamiento correcto del sistema, tanto desde el punto de vista funcional como de eficiencia.

En principio, la concurrencia comenzó a ser estudiada en el marco del diseño de los sistemas operativos, donde surgen diversas situaciones en las que es necesario proteger un recurso compartido contra accesos concurrentes indiscriminados. Posteriormente, la aparición de los sistemas de tiempo real y de los sistemas distribuidos ha motivado que éste sea un campo de intensa investigación en la actualidad.

Ahora bien, la enorme complicación de estos sistemas concurrentes ha hecho imprescindible el desarrollo de modelos formales que permitan estudiar con rigor, y por tanto con la consiguiente seguridad que ello comporta, la corrección de los sistemas diseñados. Los modelos introducidos se basan todos ellos en el concepto de abstracción, en el sentido de que todo modelo debe capturar únicamente aquellas

características del sistema que se consideren importantes. En el caso del modelado de sistemas concurrentes abstraemos gran parte de las acciones desarrolladas por los procesos, para centrarnos, especialmente, en la cooperación entre los procesos, que es el punto de máxima dificultad en el diseño de sistemas concurrentes. Una vez construido el modelo, es necesario disponer de herramientas que nos permitan analizar determinadas propiedades de "buena conducta" del mismo. Las propiedades a estudiar se dividen en dos clases: por una parte tenemos las propiedades de seguridad (safeness), que garantizan que el sistema no alcanzará nunca un estado no deseado; y por la otra propiedades de actividad (liveness), que garantizan que el sistema, con independencia del estado en el que se encuentre, alcanzará eventualmente un determinado estado de un cierto conjunto de estados.

Dentro de los modelos formales introducidos distinguiremos, por un lado los modelos algebraicos como CCS [Mil80, Mil89], CSP [Hoa78, Hoa85], LOTOS [BB87]. Estos modelos se basan en un lenguaje estructurado con una serie de operadores, cada uno de los cuales expresa una forma distinta de construir un sistema a partir de otro u otros más simples.

Por otro lado se encuentran los modelos basados en "representaciones gráficas", que expresan a un nivel espacial el paralelismo del sistema. Entre ellos podemos destacar los autómatas, y las Redes de Petri [Pet62] que son sin duda el modelo más divulgado y estudiado.

En todas las versiones básicas de estos modelos se parte, de una forma u otra, de la idea de que un sistema concurrente evoluciona mediante la ejecución en cada momento por parte de una de sus componentes (o de todas aquellas que deban sincronizar al efecto de hacerlo posible) de una acción de entre las que, en ese momento, sea capaz de ejecutar. Ello liga inmediatamente la concurrencia con el no–determinismo, pues si en un momento son varias las acciones que podrían ejecutarse, no se toma partido por ninguna de ellas, sino que se considera que todas ellas son igualmente posibles, si bien finalmente sólo uno de los posibles funcionamientos se producirá en cada ejecución del sistema.

1.1.1. Importancia del tiempo en las especificaciones

Tal y como hemos indicado anteriormente, la aparición de los sistemas de tiempo real ha originado que el factor tiempo, que antes era abstraído en los modelos, pase a

ser considerado como un elemento especialmente importante de los mismos. Evidentemente, si debe diseñarse un sistema que sea capaz de responder a un cierto estímulo externo en un plazo de tiempo limitado, está claro que el modelo que utilicemos debe ser capaz de describir el tiempo a un nivel cuantitativo, y no sólo cualitativo, para así poder determinar si el diseño realizado satisface los requerimientos impuestos. Otro ejemplo igualmente interesante lo encontramos en el diseño de protocolos de comunicación, que constituyen uno de los productos software de mayor complejidad en su desarrollo. Los problemas que encontramos en el diseño de protocolos se relacionan con la necesidad de garantizar una fiabilidad superior a la que habitualmente proporciona una vía de comunicación, lo que conlleva la aparición dentro del diseño de time-outs, que permiten la reemisión de un mensaje cuando se detecta, pasado un cierto tiempo, que no se ha recibido el preceptivo acuse de recibo del receptor del mismo.

En consecuencia, han aparecido diversas extensiones de los modelos clásicos que incorporan el tiempo como un factor más de diseño. Si nos centramos en primer lugar en los modelos algebraicos, podemos comenzar citando a Reed y Roscoe, que han definido una extensión temporizada de CSP [RR88] manteniendo los principios básicos del CSP clásico. En este modelo la ejecución de acciones no toma tiempo, pero se incluye: un nuevo operador (delay) que permite modelar el paso del tiempo. Además, para evitar que se dé el fenómeno irreal de la ejecución en un período finito de tiempo de infinitas acciones, es necesario asumir un retraso mínimo entre la ejecución secuencial de cada par de acciones. Moller y Tofts [MT90] han definido una extensión temporizada del CCS, TCCS, donde de nuevo las acciones no toman tiempo y se crean nuevos operadores para diseñar el paso del tiempo, así como un nuevo operador de elección, llamado elección débil. Una aproximación diferente ha sido adoptada en [Ort90], donde se define una extensión temporizada de CSP, TCSP, basada en la idea de asociar duraciones a las acciones.

También, Bolognesi, Lucidi y Trigila en [BLT90] definen dos modelos que extienden LOTOS con el factor tiempo. En primer lugar definen *LOTOS con Acciones Temporizadas*, que se encuentra inspirado en el modelo de redes de Petri con Arcos temporizados. Dado el problema de expresividad que plantean estos modelos, los cuales no permiten modelar adecuadamente el escenario de "esperar hasta", definen un nuevo modelo, *LOTOS con interacciones temporizadas*, el cual se basa en las redes de Petri con transiciones temporizadas.

Dentro de los modelos gráficos ha ocurrido lo mismo y así han aparecido diversas

extensiones. En el entorno de los autómatas, aparece el modelo introducido por Alur y Dill [AD90], el cual es obtenido añadiendo un conjunto de variables reales llamadas relojes al modelo de autómatas finitos clásicos. Estos relojes se incrementan de forma continua y todos ellos a la misma velocidad. El valor de estos relojes es comprobado por las transiciones y en algunos casos son reinicializados a cero. La comprobación consiste en la comparación de su valor, o de la diferencia de dos de ellos con una constante.

En el marco de las redes de Petri, se han realizado diferentes aproximaciones para la introducción de aspectos temporales. En [Bow96] podemos encontrar un estudio sobre estas diferentes aproximaciones. Entre éstas, existe un primer grupo en el que se asocia una duración a cada una de las transiciones, bien mediante un valor fijo como se hace en [Sif77, Ram74] o bien mediante una distribución de probabilidades como se hace en [AMBB$^+$85]. Otros modelos, como el presentado en [Mer74], consideran que las transiciones no toman tiempo en su disparo, pero asocian a cada transición de la red dos números enteros, que representan, respectivamente, el primer instante en el que la transición puede ser disparada, y el instante final en el que ésta debe ser disparada, ambos períodos de tiempo contados desde el momento en el que la transición comienza a estar permitida. Finalmente encontramos otros modelos [vdA93, vdAO95] los cuales incluyen tiempo en los tokens. Más recientemente, Cerone y Maggiolo-Schettini [CMS99] han propuesto un modelo más general, al que denominan Redes de Petri Temporizadas estáticamente, donde las restricciones temporales son intervalos asociados estáticamente a los lugares, transiciones y arcos.

Citaremos por último el modelo de *Redes de Petri con Arcos Temporizados* introducido en [Wal83], que está basado en la asociación de un intervalo de tiempo a cada arco de la red de la forma (p,t). En dicho modelo los tokens sobre los lugares tienen también asociada una edad, la cual es 0 cuando éstos son generados, y va incrementándose con el paso del tiempo, hasta que el token sea consumido por el disparo de alguna transición. Una transición sólo puede dispararse cuando cada lugar precondición tenga un conjunto suficiente de tokens válidos, que son aquéllos cuya edad está comprendida en el intervalo asociado al arco que conecta el lugar con la transición en cuestión. Aparecen así tokens muertos, cuya edad ya no les permite intervenir en el disparo de ninguna transición, pudiendo ser por tanto ignorados en lo sucesivo.

1.1.2. Introducción de probabilidades

Otras ampliaciones de los modelos clásicos se centran en la introducción en el modelo de informaciones probabilísticas que permitan cuantificar ciertas decisiones que aparecen en los procesos, permitiendo al mismo tiempo que otro tipo de decisiones sigan resolviéndose de forma no-determinista. La principal novedad y ventaja que presenta este enfoque probabilístico/no–determinista frente al probabilístico puro es que, mientras en este último es necesario cuantificar probabilísticamente todos los comportamientos del sistema, en el primero de ellos se cuantifican sólo aquellos comportamientos susceptibles de ser bien conocidos, mientras que se sigue utilizando el no–determinismo para modelar aquellos otros comportamientos cuya cuantificación es difícil de conocer o incluso imposible.

La primera referencia de la que tenemos constancia en la que se discute un modelo algebraico de procesos concurrentes en el que se incorporan aspectos cuantitativos probabilísticos es [LS89], donde la aproximación a los procesos concurrentes se hace en base a las transiciones operacionales que generan, al margen de una sintaxis concreta para los mismos. En dicho trabajo se extiende la noción de sistemas de transiciones etiquetadas introduciendo informaciones probabilísticas, considerando una interpretación *reactiva* de las probabilidades. Se extiende la noción de bisimulación, definiendo una nueva relación de equivalencia entre procesos probabilísticos: la bisimulación probabilística. La noción de bisimulación probabilística es mucho más fuerte que la de bisimulación de Milner [Mil89] ya que se exige no sólo que los procesos deriven las mismas clases de equivalencia, sino que además debe hacerse con exactamente la misma probabilidad.

En [vGSST90, vGSS95] se realiza un análisis que permite clasificar y caracterizar de qué forma puede llevarse a cabo la inclusión de probabilidades en un modelo. Así se presentan los modelos reactivo, generativo y estratificado para procesos concurrentes probabilísticos.

En [CdFV96, CdFV97], se presenta una extensión probabilística de CSP, que reemplaza la elección interna por una elección probabilística generativa, y la externa por una elección probabilística reactiva. Se estudian tres semánticas para el nuevo lenguaje: de pruebas, denotacional y operacional, las cuales se prueba que son equivalentes. Además, se da un sistema de prueba que es correcto y completo respecto a las semánticas anteriores.

Han aparecido también diversos trabajos en los que se han extendido los modelos

clásicos introduciendo en ellos ambos factores, tiempo y probabilidad, aunque estas extensiones han sido estudiadas en menor medida. Entre los trabajos que han abordado este tema podemos destacar [HJ90] donde los autores presentan TPCCS, una extensión de CCS que permite la especificación de sistemas asíncronos con probabilidades y tiempos. Para ello introduce un nuevo operador de elección probabilística y dos operadores, uno a nivel sintáctico que denomina "time-out", y otro a nivel semántico que representa el paso del tiempo.

1.2. De los modelos algebraicos a los gráficos

La definición de los modelos formales para el desarrollo de sistemas concurrentes, como puede constatarse en lo expuesto anteriormente, ha discurrido por dos vías diferenciadas, lo que ha hecho que aparezcan dos grandes grupos de modelos, por un lado los modelos algebraicos y por otro los gráficos, cada uno de los cuales posee sus ventajas y sus inconvenientes.

Entre las ventajas que poseen las álgebras de procesos podemos destacar su proximidad a los lenguajes de programación, lo que hace que la especificación del comportamiento de los sistemas concurrentes se realice de una forma fácil e intuitiva. Estas características hacen que los diseñadores de software trabajen más a gusto con ellas que con otros modelos.

Por otro lado, entre las ventajas de los modelos gráficos podemos destacar que poseen una interpretación fácil e intuitiva debido a su naturaleza gráfica, así como la existencia de unas técnicas de verificación basadas en estos modelos cuya aplicación es más cómoda y menos tediosa que los métodos definidos para las álgebras de procesos.

Con el fin de conseguir aprovechar las ventajas de ambos tipos, algunos investigadores trabajan en la integración de ambos modelos, la cual se hace principalmente mediante la definición de métodos de traducción de un modelos a otro. Concretamente la traducción se realiza desde los modelos algebraicos a los gráficos.

En esta línea, Nicollin, Sifakis, y Yovine en [NSY93, NSY92] presentan el álgebra ATP (Algebra of Timed Processes) [NS90], la cual posee diversos operadores que permiten la especificación de comportamientos temporales. Entre ellos podemos destacar el operador de time-out y de retraso, pero no posee un operador específico

de prefijo de acción urgente. Tras definir el álgebra presenta la traducción de los operadores de la misma a grafos temporizados, los cuales son autómatas temporizados del tipo definido en [AD90]. En estos grafos, cada transición posee un conjunto de restricciones asociado, el cual debe cumplirse para que esa transición pueda ser disparada. Así mismo, la ejecución de una transición conlleva que algunos de los relojes del sistema sean reseteados a 0. D'Argenio y Brinksma, en [DB96] definen un nuevo lenguaje algebraico para el que posteriormente definen su traducción a autómatas temporizados. Los autómatas utilizados en este trabajo, a diferencia de los utilizados en los trabajos enumerados anteriormente, realizan el reseteo de los relojes cuando se alcanza un nuevo nodo del autómata y no cuando se ejecuta una transición. Así, el conjunto de relojes a resetear se encuentra asociado a los nodos y no a las transiciones.

Las traducciones de las álgebras de procesos a redes de Petri datan de finales de los 80. Así, Goltz [Gol88] presenta una traducción de CCS a redes de Petri, mientras que Taubner [Tau89] define una traducción para un álgebra más general.

Más recientemente, Best, Devillers y Hall han definido el Petri Box Calculus, [BDH92], el cual toma los operadores clásicos de las álgebras de procesos y los aplica a redes de Petri, con lo que se consigue un comportamiento composicional de las redes resultantes.

1.3. Motivación y justificación

Tradicionalmente, los diseñadores de sistemas concurrentes, antes de iniciar su trabajo, seleccionaban uno de los formalismos para ser utilizado y se centraban en ése, descartando el resto de modelos. Con esto se estaban descartando las ventajas que los otros modelos podrían aportar a su trabajo.

Ante esta situación, parece interesante tener una relación definida entre los diferentes modelos, de modo que una vez realizada una especificación en uno de ellos, concretamente en un modelo algebraico, ésta pueda ser transformada a otro de los modelos, dependiendo de las necesidades del momento, y de las facilidades que cada uno de los modelos proporcione para el estudio de unas determinadas propiedades.

Con esta idea en la cabeza, se inició el trabajo que ha culminado en esta tesis. En primer lugar se definió el modelo algebraico base, el cual se decidió que fuese un

modelo temporizado y probabilístico. Este modelo está basado en LOTOS, añadiendo al mismo operadores probabilísticos para la especificación de sistemas en los que su comportamiento se defina en función de unas probabilidades conocidas, y algunos operadores temporizados, que permitiesen la especificación de los sistemas cuyo comportamiento esté muy ligado a restricciones temporales como son los sistemas de tiempo real. Con este nuevo modelo se intentaron recoger las mejores características de algunos de los modelos existentes en un único modelo.

Tras definir el modelo algebraico base, el primer modelo al que se tradujo este álgebra fue el modelo de grafos de estados dinámicos. Este modelo está basado en los autómatas temporizados definidos en [AD90], aunque posee algunas diferencias con respecto a los autómatas entre las cuales podemos destacar el uso de los relojes, que en nuestro caso no serán reseteados a 0 por ninguna transición, sino que éstos son sincronizados con el instante de tiempo en que se ejecuta la acción. Ello nos permite conocer de una forma inmediata el tiempo consumido por cada componente de la especificación, aunque tiene el inconveniente de dificultar el proceso de verificación formal. Otra diferencia importante es la forma de tratar la urgencia, o los retrasos acotados, que en este modelo se capturan mediante acciones especiales (τ), o mediante el operador *wait*, mientras que en los autómatas temporizados se hace por medio de los invariantes.

Tras definir el nuevo modelo, definimos la traducción de los términos del álgebra a este nuevo modelo. Esta traducción se realiza siguiendo las ideas planteadas en [NSY93, NSY92], aunque las diferencias entre el trabajo presentado en esta tesis y el presentado en aquél son sustanciales. Entre ellas podemos destacar aquellas que provienen del hecho de la existencia de operadores diferentes en ambos modelos algebraicos y en especial de la existencia de los operadores probabilísticos en nuestro modelo. Otras diferencias en el método de traducción provienen de las diferencias existentes entre el modelo de grafos de estados dinámicos y los autómatas (algunas ya mencionadas, pero para más detalles ver capítulo 4). En la traducción a Redes de Petri hemos realizado una extensión temporizada de los resultados mostrados en [Gol88] y [Tau89], si bien hemos planteado además posibles extensiones de la traducción clásica explotando las características particulares del modelo de red de Petri temporizada que hemos utilizado las *redes de Petri con arcos temporizados*.

1.4. Resumen de la tesis

El resto de capítulos de esta tesis desarrollan las ideas expuestas anteriormente, siguiendo el orden siguiente.

Capítulo 2. Generalidades

En este capítulo presentaremos los conceptos fundamentales relacionados con los modelos utilizados en esta tesis, concretamente los relacionados con las álgebras de procesos, los grafos y autómatas temporizados, y las redes de Petri.

Capítulo 3. Álgebra Temporizada (TPAL)

El álgebra *TPAL*, presentada en este capítulo, está basada en el lenguaje LOTOS [BB87], al cual se le han añadido nuevos operadores para la especificación de comportamientos temporales.

Posteriormente definimos la semántica operacional de TPAL utilizando el estilo estructurado de Plotkin [Plo81] y Milner [Mil89], por medio de un sistema de transiciones etiquetado, definido a partir de una colección de reglas que capturan el comportamiento de cada uno de los operadores.

Este álgebra será posteriormente ampliada con unos operadores probabilísticos en el capítulo 7.

Capítulo 4. Grafos de Estados Dinámicos

En este capítulo se realiza la definición del modelo de grafos de estados dinámicos, y se proporciona una traducción de los términos del álgebra temporizada a este modelo de grafos.

Esta traducción no será aplicable a todos los términos del álgebra, sino que será necesario introducir restricciones sintácticas sobre los mismos, concretamente la traducción será aplicable a los denominados términos regulares.

Tras la definición formal de la traducción presentaremos dos teoremas, que establecen la equivalencia entre ambos modelos semánticos.

Con el fin de paliar el problema de la explosión de estados que presentan los grafos de estados dinámicos, presentaremos también en este capítulo algunos algoritmos, que aplicados a los grafos obtenidos, nos permitirán simplificar y reducir los mismos.

Para formalizar estos algoritmos definiremos previamente una relación de equivalencia entre los grafos.

Capítulo 5. Relación con los Autómatas Temporizados

En este capítulo estableceremos la relación de los grafos de estados dinámicos con los autómatas temporizados, mostrando la traducción de los mismos a autómatas temporizados [AD90]. Con respecto a la traducción inversa, aunque no podemos pronunciarnos, nuestra opinión es que los grafos son más débiles que los autómatas temporizados, si bien no hemos logrado probarlo formalmente.

Capítulo 6. Traducción del Álgebra a Redes de Petri

En este capítulo se realizará la traducción de los términos del álgebra a un modelo de redes de Petri, concretamente a redes de Petri con arcos temporizados [BLT90, Wal83, Han93, VdFC99, AN01]. En este modelo, los tokens tienen asociado un valor real no negativo, que indica el tiempo transcurrido desde su creación (su edad); y los arcos que van de un lugar a una transición estarán etiquetados por intervalos de tiempo, los cuales establecen restricciones sobre la edad de los tokens que pueden ser usados para disparar la transición. Como consecuencia de estas restricciones algunos tokens pueden convertirse en muertos, ya que nunca estarán disponibles para ser usados, debido a que son demasiado viejos y no permitirán el disparo de ninguna transición en el futuro.

Capítulo 7. TPPAL. Extensión del Álgebra TPAL con Probabilidades

En este capítulo realizaremos una extensión del álgebra definida en el capítulo 3 con unos operadores probabilísticos. Introduciremos un nuevo conjunto de reglas probabilísticas para adecuar la semántica operacional, y será necesario modificar algunas de las introducidas para los términos temporizados, de modo que se capture de forma correcta la evolución de todos los procesos.

Capítulo 8 Grafos de Estados Dinámicos Probabilísticos

Se definirán como una extensión probabilística de los grafos de estados dinámicos incluyendo las probabilidades. Mostraremos las ideas clave de la traducción, así como los teoremas de equivalencia.

Capítulo 9. TPAL, Una Herramienta de Apoyo al Diseño de Sistemas Concurrentes

Las definiciones formales de los diferentes modelos, y las traducciones antes indicadas son el marco formal sobre el que se apoya la herramienta TPAL que nuestro grupo de investigación está desarrollando desde 1997, cuya evolución ha ido muy ligada a la de esta tesis. En TPAL, se parte de una especificación algebraica, a partir de la cual se generan diversos modelos gráficos como pueden ser grafos de estados dinámicos, redes de Petri con arcos temporizados y autómatas temporizados. Estos modelos nos permitirán verificar propiedades de comportamiento del sistema, así como realizar una simulación de la ejecución del sistema especificado.

En este capítulo presentamos las principales características de la herramienta.

Capítulo 10. Conclusiones y Trabajos Futuros

Finalmente, en este capítulo presentaremos nuestras conclusiones, así como algunas líneas de trabajo futuro, las cuales irán fundamentalmente orientadas a la verificación de los modelos, y a completar las implementaciones de la herramienta.

Capítulo 2

Generalidades

Este capítulo está dedicado a presentar los conceptos fundamentales de los diferentes modelos relacionados con esta tesis. Concretamente, se realiza una breve introducción a la sintaxis de LOTOS, como base del álgebra aquí definida. También se exponen los conceptos básicos sobre autómatas temporizados, y sobre redes de Petri con arcos temporizados.

Con ello, se pretende proporcionar la base de trabajo para los capítulos siguientes de la tesis.

2.1. LOTOS (Language of Temporal Ordering Specifications)

LOTOS es un lenguaje de especificación con una base formal, diseñado inicialmente para la especificación de sistemas OSI, pero que es igualmente válido para la especificación de sistemas concurrentes y distribuidos.

El lenguaje está compuesto por un álgebra de procesos, basada en CCS y CSP, la cual permite la descripción de comportamientos, y por un lenguaje algebraico (ACT-ONE [EM85]), para la descripción de tipos abstractos de datos.

Permite la descripción de concurrencia, no-determinismo, y comunicaciones síncronas y asíncronas. Soporta varios niveles de abstracción y provee varios estilos de especificación.

Podemos considerar que un modelo LOTOS de un sistema es una caja negra

la cual posee un número determinado de puertas que pueden ser vistas desde el entorno, y que representan los eventos o acciones que puede realizar el sistema. De este modo el sistema funciona mediante la activación de esas puertas.

Entre los eventos que proporciona un sistema se encuentran los eventos de entrada/salida, para los cuales es necesario indicar en qué puerta se producen y qué datos serán trasferidos. Así, para los eventos de entrada se indicará la variable sobre la que será recogido el dato y el tipo de éste. Para represetar un evento de entrada se utiliza el formato siguiente:

$$< puerta > ? \ < variable >: tipo$$

Para indicar un evento de salida se utiliza la expresión siguiente:

$$< puerta > ! \, expresion$$

donde, como puede observarse, se indica en qué puerta se producirá la salida de datos y qué dato será el proporcionado.

Junto a esta forma de representar las acciones de un sistema, LOTOS posee los operadores siguientes para la descripción del comportamiento de un sistema:

Terminación. La terminación de un proceso puede producirse mediante la ejecución de la acción *stop* o de la acción *exit*. El proceso *stop*, como es habitual, representa un proceso que no permite realizar ninguna acción, mientras que el proceso *exit* representa una terminación satisfactoria. Este proceso realiza una acción especial, la cual es representada por el símbolo "¡"(admiración), y después se comporta como stop. La acción ¡ forma parte del modelo matemático de LOTOS, no siendo una acción que pueda ser usada explícitamente dentro de una especificación.

Secuencia. Representada por el símbolo ";"(punto y coma), se emplea para prefijar un comportamiento o proceso con una acción. La sintaxis es $acc; P$.

Habilitación $P >> P'$. Es una generalización del operador secuencia, que combina dos comportamientos en uno. Si el proceso P termina con éxito, entonces se podrá producir el proceso P'; pero si P no termina satisfactoriamente, el proceso P' no ocurrirá.

Deshabilitación $P[> P'$. Este operador se introduce en el lenguaje ante la necesidad que aparece, en muchas situaciones, de especificar comportamientos que

son interrumpidos por algo. Si el proceso P' ejecuta alguna acción, entonces interrumpe la ejecución del proceso P pasando el sistema a comportarse, a partir de ese momento, como el proceso P'. Si P' no ejecuta ninguna acción, el sistema se comportará como P y si éste termina satisfactoriamente, el sistema completo finalizará satisfactoriamente.

Elección $P[]P'$. Es el operador clásico de elección externa.

Cada uno de los procesos ofrece al entorno la acción que puede realizar, siendo el entorno, en función de sus necesidades, el que elija la acción que se ejecutará. Si ambos procesos ofrecen la misma acción, entonces la elección de qué proceso la ejecuta será realizada internamente por el sistema.

Paralelismo. La composición paralela de dos procesos en LOTOS puede ser especificada mediante dos operadores diferentes.

El primero, $P|||P'$, representa la evolución entrelazada de los procesos P y P', sin ninguna acción de sincronización.

El segundo es $P|[A]|P'$, que indica la ejecución en paralelo de los procesos P y P', sincronizando en el conjunto de acciones A. Si no se indica un conjunto de acciones de sincronización, entonces se entiende que ambos procesos sincronizan en todas las acciones que aparecen en ambos procesos.

Ocultamiento. Representado sintácticamente como *hide a in P*. Oculta la ocurrencia de la acción a dentro del proceso P.

Acción interna. Se representa como i. Esta acción se ejecuta internamente por el sistema, sin influencia del entorno del mismo. Al combinar el operador *exit* con el operador de habilitación, la acción ¡ se transforma en la operación i.

Junto a este lenguaje para la descripción de procesos, LOTOS, como se comentó al principio, posee un sublenguaje de tipos de datos, ACT-ONE, que permite definir variables dentro de las especificaciones, así como nuevos tipos abstractos de datos. Las variables son variables matemáticas, es decir, son solamente nombres a los que se asignan valores en concreto. El comportamiento de los operadores correspondientes a los tipos de datos se define mediante una semántica ecuacional.

En los últimos años este lenguaje ha sido sometido a una revisión por ISO dando lugar a E-LOTOS [LOT96a, LOT96b, LOT98]. Esta revisión y modificación ha consistido básicamente en la introducción de aspectos temporales.

2.2. Autómatas finitos

Habitualmente el comportamiento de un sistema se define mediante el conjunto de secuencias de acciones que puede ejecutar ese sistema. Desde el punto de vista formal podemos considerar que un conjunto de secuencias es un lenguaje formal, y la forma habitual de representar los lenguajes formales es mediante los autómatas. Por tanto, podemos decir que los autómatas son una buena herramienta para la descripción del comportamiento de un sistema. Entre los diferentes modelos de autómatas que existen, los más utilizados son los autómatas finitos ya que la mayor parte de los sistemas estudiados poseen un número finito de estados.

Con el paso del tiempo, los autómatas finitos se han convertido en el estándar usado para la verificación de sistemas concurrentes, sobre todo en la aplicación de técnicas de model-checking.

En esta sección introduciremos los conceptos relacionados con los autómatas finitos, y en la siguiente expondremos los conceptos relacionados con los autómatas temporizados.

Definición 1 (Tabla de transición)
Una *tabla de transiciones* A es una tupla:

$$(Q, \Sigma, E, Q_0)$$

donde

- Q es un conjunto finito de estados. Sus elementos suelen denotarse habitualmente con las letras $p,\ q,\ \ldots$.
- Σ es un alfabeto finito de entradas, cuyos elementos suelen denotarse por letras $a,\ b,\ \ldots$.
- $Q_0 \subseteq Q$ es el conjunto de estados iniciales del sistema.
- $E \subseteq Q \times \Sigma \times Q$ es el conjunto de arcos. La tupla (q_i, a, q_j) representa un arco desde el nodo q_i al nodo q_j, etiquetado con la acción a.

□

Habitualmente, una tabla de transiciones se representa gráficamente mediante un grafo dirigido, donde los vértices corresponden a los estados, y los arcos corresponden a las transiciones.

Ejemplo 1 Consideremos la tabla $A = (Q, \Sigma, E, Q_0)$, definida por los conjuntos:

$$\begin{aligned} Q = &\quad \{q_0, q_1, q_2, q_3, q_4\} \\ \Sigma = &\quad \{a, b\} \\ Q_0 = &\quad \{q_0\} \\ E = &\quad \{(q_0, a, q_0), (q_0, b, q_4), (q_1, a, q_1), (q_1, b, q_2), (q_2, a, q_2), \\ &\quad (q_2, b, q_2), (q_3, a, q_3), (q_3, b, q_2), (q_4, a, q_0), (q_4, b, q_2)\} \end{aligned}$$

Su representación gráfica será la mostrada en la Figura 2.1.

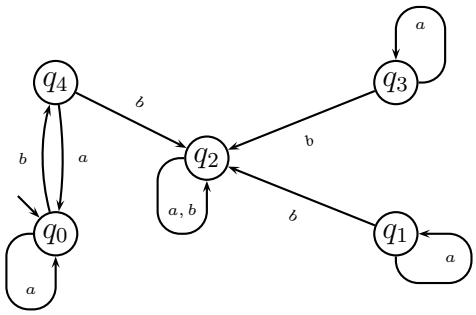


Figura 2.1: Tabla de transiciones

□

Definición 2 (Computación)
Sea $A = (Q, \Sigma, E, Q_0)$ una tabla de transiciones, y $\sigma = \sigma_1\ \sigma_2\ \sigma_3\ \ldots$ una palabra sobre el conjunto Σ.

La secuencia $r : s_0 \xrightarrow{\sigma_1} s_1 \xrightarrow{\sigma_2} s_2 \xrightarrow{\sigma_3} \ldots$ será una computación de A si:

- $s_0 \in Q_0$
- $(s_{i-1}, \sigma_i, s_i) \in E \quad \forall\, i \geq 1$

□

Definición 3 (Tabla de transiciones determinista)
Una tabla de transiciones $A = (Q, \Sigma, E, Q_0)$ es *determinista* si cumple:

- $|Q_0| = 1$, es decir, sólo existe un estado inicial.
- Para cada estado del sistema y cada acción de Σ, existe como mucho un arco saliente de dicho estado etiquetado con dicha acción.

□

Para una tabla determinista, dada una palabra, existirá como máximo una computación para la misma.

Es habitual añadir a las tablas de transiciones una condición de aceptación de palabras finitas, convirtiéndose en tal caso en *autómatas*. Si estos autómatas aceptan palabras infinitas, entonces se denominarán ω-autómatas ([Buc62, Cho74, McN66, Var87]).

Dada la estrecha relación existente entre las tablas de transiciones y los autómatas, en la literatura sobre la aplicación de estos modelos a la especificación de sistemas concurrentes, se abusa de la notación y se denomina *autómata* a lo que realmente es una *tabla de transiciones*. Nosotros, en esta tesis, haremos también uso de esta notación y denominaremos autómatas a las tablas de transición.

2.3. Autómatas temporizados

El gran auge que han tenido los sistemas con restricciones temporales ha hecho que haya sido necesario extender los autómatas finitos, de modo que permitan capturar estas restricciones temporales.

Un grupo de estas extensiones tratan el tiempo mediante valores continuos [HNSY, LPY95, SB, YS], aunque sin duda el modelo que se ha convertido en el estándar para el estudio de sistemas con tiempo continuo es el modelo de autómatas temporizados definido por Alur, Courcoubetis y Dill en su trabajo [ACD90, AD90].

Los autómatas temporizados son autómatas finitos a los que se les ha añadido un conjunto de variables reales, llamadas relojes. Estos relojes se emplean para medir el paso del tiempo.

Como paso previo a la definición de un autómata temporizado, definiremos lo que es un reloj, y lo que se entiende por restricción temporal.

Definición 4 (Reloj)
Un *reloj* es una variable que toma valores sobre $\mathbb{R}_0^+$.

□

Definición 5 (Restricciones temporales)
Sea C un conjunto de relojes cuyos elementos representaremos como R_i.

Las restricciones temporales sobre C, $\Phi(C)$, se definen como sigue:

$$\alpha ::= R_i \prec c \mid R_i - R_j \prec c \mid \neg\alpha \mid (\alpha \wedge \alpha)$$

siendo $c \in \mathbb{R}_0^+$ y $\prec \in \{ <, \leqslant \}$.

□

Definición 6 (Autómata temporizado)[ACD90, Kat98]

Definimos un *autómata temporizado* como una tupla:

$$A = (Q, \Sigma, Q_0, E, clocks, times, guard, inv)$$

donde

(Q, Σ, Q_0, E) es una tabla de transiciones.

clocks es un conjunto finito de relojes.

times: $E \longrightarrow 2^{clocks}$ es una función que asigna a cada arco e un conjunto de relojes $times(e)$. Este conjunto de relojes será reseteado cuando esa transición sea ejecutada.

guard: $E \longrightarrow \Phi(clocks)$ es una función que etiqueta los arcos con un conjunto de restricciones temporales, las cuales llamaremos guardas. Estas restricciones deben cumplirse para que la transición pueda ser ejecutada.

inv: $Q \longrightarrow \Phi(clocks)$ es una función que asigna a cada nodo n del autómata un conjunto de restricciones, llamadas invariantes, que establecen unas condiciones que deben cumplirse mientras nos encontremos en ese nodo.

□

Ejemplo 2 En la Figura 2.2 podemos ver un autómata temporizado, en el cual el nodo s_1 tiene asociado un invariante, el arco s_1 a s_2 posee una guarda y un conjunto de relojes a resetear, el arco s_2 a s_0 solamente tiene un conjunto de relojes a resetear, y existen otros arcos que no poseen ni guardas ni conjunto de relojes a resetear.

□

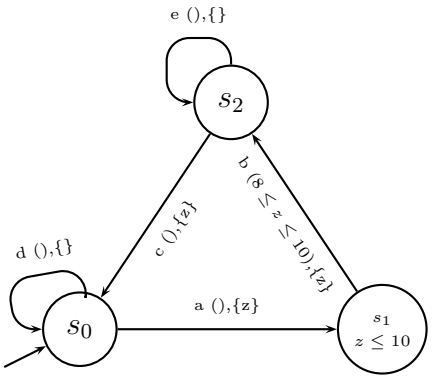


Figura 2.2: Autómata temporizado

Definición 7 (Evaluación de relojes)
Dado un autómata temporizado $A = (Q, \Sigma, Q_0, E, clocks, times, guard, inv)$.

Una *evaluación* v del conjunto de relojes $clocks$ es una función $v : clocks \longrightarrow \mathbb{R}_0^+$ que asigna a cada reloj $x \in clocks$ su valor actual $v(x)$.

Denotaremos como $V(clocks)$ el conjunto de todas las evaluaciones sobre $clocks$.

□

Definición 8 (Estado)
Dado un autómata $A = (Q, \Sigma, Q_0, E, clocks, times, guard, inv)$. Definimos un *estado* del mismo como un par (q, v) donde $q \in Q$ y $v \in V(clocks)$.

□

2.3.1. Semántica

La interpretación de un autómata temporizado se define en términos de un sistema de transiciones infinito $(S, \longrightarrow)$, donde S es un conjunto de estados, y $\longrightarrow$ es la relación de transición que define como se evoluciona de un estado a otro.

Un autómata temporizado puede evolucionar de dos maneras, ejecutando una transición o mediante el paso del tiempo (permaneciendo en el mismo nodo), aunque ambas formas son representadas mediante una relación de transición $\longrightarrow$.

Definición 9 (Sistema de transiciones) [Kat98]
Sea un autómata temporizado $A = (Q, \Sigma, Q_0, E, clocks, times, guard, inv)$.

El *sistema de transiciones* asociado a A, que representaremos como $M(A)$ viene definido por la tupla:

$$(S, s_0, \longrightarrow)$$

donde

- $S = \{(q, v) \in Q \times V(clocks) \mid v \models inv(q)\}$
- $s_0 = (q_0, v_0)$ donde $v_0(x) = 0 \; \forall \, x \in clocks$
- La relación de transición $\longrightarrow \subseteq S \times (\mathbb{R}_0^+ \cup \{*\}) \times S$ se define por las reglas siguientes:

  1. $(q, v) \xrightarrow{*} (q',$ reset $times(e)$ in $v)$ si se cumplen las condiciones siguientes:
     - *a)* $e = (q, q') \in E$
     - *b)* $v \models guard(e)$
     - *c)* (reset $times(e)$ in $v) \models inv(q')$
  2. $(q, v) \xrightarrow{d} (q, v + d)$ para un valor real positivo d si se cumple la condición
     - *a)* $\forall d' \leqslant d \;\; v + d' \models inv(q)$

  donde reset $times(e)$ in v representa una evaluación v de los relojes en la cual aquellos relojes que etiquetan la transición e han sido reseteados a 0 y el resto poseen los valores que tenían en la evaluación v, es decir:

$$\forall \, x \in clocks, \quad \text{reset } times(e) \text{ in } v = \begin{cases} 0 & \text{si } x \in times(e) \\ v(x) & \text{si } x \notin times(e) \end{cases}$$

  $v + d$ es una nueva evaluación de los relojes del grafo en la cual el valor de cada uno de los relojes ha sido incrementado en d unidades, es decir:

$$\forall \, x \in clocks, \;\; (v + d)(x) = v(x) + d$$

□

El conjunto de estados de A contendrá todos aquellos estados (q, v) en los cuales los valores de v no invaliden el invariante de q. Entre esos estados pueden existir algunos inalcanzables.

La relación de transición establece las posibles evoluciones del autómata.

Las evoluciones pueden realizarse mediante el disparo de alguna transición, lo cual viene recogido por la claúsula 1. Para que esta evolución se produzca debe pasarse de un nodo a otro a través del arco e, para lo cual los valores v deben hacer cierta la guarda de la transición e, y los nuevos valores de los relojes, obtenidos mediante el reseteo de los relojes asociados al arco e, deberán satisfacer el invariante del nodo de destino q'.

También se puede evolucionar por el paso del tiempo, lo que es capturado con la claúsula 2, en la cual se establece que para permitir el paso de d unidades de tiempo debe cumplirse el invariante del nodo q en todos los instantes entre 0 y d.

Definición 10 (Camino)
Un *camino* es una secuencia infinita $s_0\, a_0\, s_1\, a_1\, s_2\ a_2\ \ldots$ compuesta por estados y etiquetas de transiciones, de modo que:

$$s_i \xrightarrow{a_i} s_{i+1} \quad \forall i \geqslant 0,\ s_i \in S,\ a_i \in \mathbb{R}_0^+ \cup \{*\}$$

Las etiquetas de las transiciones pueden ser: el símbolo $*$ si la transición se ha producido por un arco, o un número real positivo si la evolución se ha producido por una transición de paso de tiempo.

□

Definición 11 (Tiempo transcurrido en un camino)
Sea un camino $\sigma = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \ldots$ y un número $i \in \mathbb{N}$.

El *tiempo transcurrido* entre s_0 y s_i, denotado como sigue: $\Delta(\sigma, i)$ se define como

- $\Delta(\sigma, 0) = 0$
- $\Delta(\sigma, i+1) = \Delta(\sigma, i) + \begin{cases} 0 & \text{si } a_i = * \\ a_i & \text{si } a_i \in \mathbb{R}_0^+ \end{cases}$

Se dice que un camino σ es *divergente en el tiempo* si $\lim\limits_{i\to\infty} \Delta(\sigma, i) = \infty$. El conjunto de todos los caminos divergentes que parten de un estado s será denotado como $P_M^\infty(s)$.

□

Ejemplo 3 Consideremos un sistema de control automático de un paso a nivel en un cruce de trenes, cuyo funcionamiento es el siguiente:

Cuando un tren se aproxima al cruce lo comunica mediante una señal (*aprox*) al controlador del cruce al menos un minuto antes de entrar en el cruce (*in*). El tren tardará 4 minutos en cruzar tras lo cual lo indicará de nuevo al controlador mediante una señal *exit*.

El controlador espera recibir la señal de aproximación del tren. Una vez transcurrido exactamente 1 minuto después de la recepción de esta señal el controlador enviará la señal de bajada (*bajar*) a la barrera. En ese momento iniciará una espera hasta recibir la señal de que el tren ha finalizado el cruce. Transcurrido un minuto desde la recepción de la señal, indicará que la barrera puede ser elevada (*subir*).

Por otro lado, la barrera empezará a bajar una vez que reciba una señal del controlador. El proceso de bajado (*down*) durará entre 1 y 3 minutos, tras el cual la barrera estará totalmente bajada. En esta posición permanecerá hasta que de nuevo el controlador indique que puede empezar a subir. La barrera tardará en subir (*up*) entre 1 y 2 minutos.

Atendiendo a esta descripción, el autómata temporizado que modela el sistema es el mostrado en la Figura 2.3.

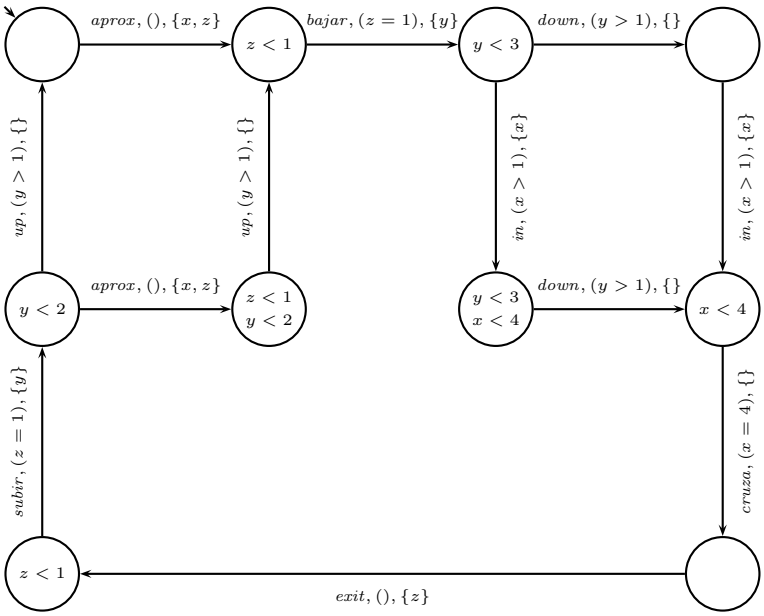


Figura 2.3: Autómata temporizado correspondiente al ejemplo 3

□

2.4. Redes de Petri

Las redes de Petri no son un modelo reciente, fueron definidas por Carl Adam Petri en 1962 [Pet62]. Es un modelo de naturaleza gráfica, aunque con un soporte formal preciso, que hace de las mismas un modelo muy útil en el diseño y análisis de sistemas concurrentes. Es precisamente esta primera característica, su simple representación gráfica, la que permite disponer de modelos de sistemas concurrentes fácilmente legibles, pues la concurrencia real de acciones queda identificada fácilmente, así como la sincronización de acciones. Por otra parte, la descripción formal de las mismas permite analizar ciertas propiedades del sistema de forma sistemática. Otra de las ventajas de este modelo es la disponibilidad de herramientas que dan soporte al diseño y análisis de sistemas con redes de Petri.

Como otros muchos modelos empleados en el proceso de diseño de sistemas informáticos, las redes de Petri permiten el uso de técnicas de refinamiento progresivo, para ir obteniendo modelos cada vez más detallados del sistema. Sin embargo, en muchos casos no resulta fácil crear un modelo de un sistema mediante redes de Petri, y es preciso acudir a otro tipo de técnicas, como las algebraicas, para posteriormente, transformar estas especificaciones en redes de Petri de forma automatizada.

La existencia de dos tipos de elementos en cualquier sistema, elementos activos (eventos o acciones) y pasivos (estados) queda reflejada en la definición formal de las redes de Petri en la que los elementos denominados *lugares* representan los elementos pasivos, mientras que los denominados *transiciones* representan a los elementos activos o acciones.

La evolución del sistema se produce mediante la ejecución de las transiciones. La posibilidad de esta ejecución está condicionada a un entorno limitado, es decir, solamente influirán en ella determinados lugares, lo que hace que dos transiciones puedan ser disparadas simultáneamente ya que sus entornos son diferentes. Esta posible simultaneidad no impone en absoluto la necesidad de hacerlas coincidir en el tiempo, sino que solamente indica que varias transiciones pueden ejecutarse a la vez pues no existe relación de causalidad entre ellas que lo impida. Este comportamiento dinámico del sistema queda modelado mediante la regla de disparo de transiciones.

Definición 12 (Red de Petri ordinaria)
Una terna $N = (P, T, F)$ formada por dos conjuntos P y T y una relación F definida sobre $P \cup T$, se dice que es una *red de Petri ordinaria* si satisface:

1. $P \cap T = \emptyset$
2. $F \subseteq (P \times T) \cup (T \times P)$
3. $dom(F) \cup cod(F) = P \cup T$

Al conjunto P se le llama *conjunto de lugares*, al conjunto T *conjunto de transiciones* y a F *relación de flujo*. F relaciona lugares con transiciones en forma de arcos entre lugares y transiciones o entre transiciones y lugares. Salvo que se indique lo contrario, supondremos que las redes con las que trabajaremos son finitas (los conjuntos P y T son finitos).

□

Las redes de Petri pueden representarse gráficamente por medio de grafos bipartitos, que son grafos que constan de dos tipos de nodos (lugares y transiciones). Los lugares se representan mediante círculos y las transiciones mediante rectángulos.

Ejemplo 4 Sea $N = (P, T, F)$ con

- $P = \{p_1, p_2, p_3\}$
- $T = \{t_1, t_2\}$
- $F = \{(p_1, t_1), (p_2, t_1), (t_1, p_3), (p_3, t_2)\}$

La representación gráfica de esta red puede verse en la Figura 2.4.

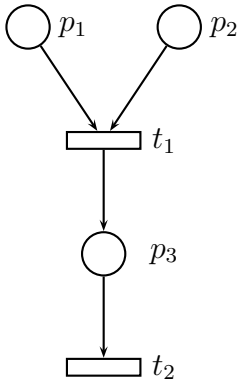


Figura 2.4: Red de Petri ordinaria

□

Definición 13 (Preconjunto y postconjunto)
Sea el conjunto $X = P \cup T$. Para todo elemento $x \in X$ definimos su *preconjunto* como:

$$^\bullet x = \{y \in X | (y, x) \in F\}$$

y su *postconjunto* como:

$$x^\bullet = \{y \in X | (x, y) \in F\}$$

□

Definición 14 (Red de Petri T-restringida)
Se dice que una red de Petri N es *T-restringida* si y sólo si $\forall\, t \in T : {}^\bullet t \neq \emptyset$ y $t^\bullet \neq \emptyset$.

□

Una red de Petri captura solamente una descripción estática del sistema que modela, no quedando reflejada en ella una descripción dinámica ni el estado en el que se encuentra el sistema en un determinado momento. Para capturar este comportamiento dinámico del sistema se introduce lo que se denomina marcado de una red.

De manera informal podemos decir que un marcado de una red representa el estado en el que se encuentra el sistema, de modo que el comportamiento dinámico de un sistema vendrá dado por la evolución del marcado mediante la aplicación de la regla de disparo, que será definida posteriormente.

Definición 15 (Marcado de una red de Petri ordinaria)
Sea $N = (P, T, F)$ una red de Petri ordinaria. Un *marcado* de N es una función $M : P \longrightarrow \mathbb{N}$.

A la tupla (P, T, F, M) se le denomina *red de Petri ordinaria marcada*.

□

Los marcados de las redes de Petri se representan gráficamente incluyendo en cada lugar tantos puntos como tokens le correspondan, o bien anotando cada lugar con el número de tokens asociado al mismo.

Ejemplo 5 Consideremos la red del ejemplo 4. Para ella consideramos el marcado siguiente:

$$M(p_1) = 1 \quad M(p_2) = 1 \quad M(p_3) = 0$$

La representación gráfica de la misma se muestra en la Figura 2.5.

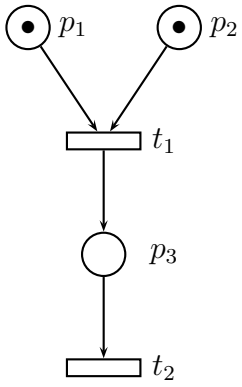


Figura 2.5: Red de Petri marcada

□

La semántica de una red de Petri marcada viene dada por una regla de disparo, que define el marcado alcanzado tras el disparo de una transición.

Definición 16 (Regla de disparo)
Sea $N = (P, T, F, M)$ una red de Petri ordinaria marcada y $t \in T$ una transición de la misma.

Se dice que la transición t *está permitida* bajo el marcado M, lo que denotaremos como $M[t\rangle$, si para todo lugar $p \in P$ tal que $(p, t) \in F$ se verifica que $M(p) > 0$.

El disparo de una transición t permitida bajo un marcado M produce un nuevo marcado M' de la red, definido de la forma siguiente:

$$M'(p) = M(p) - W_f(p, t) + W_f(t, p) \quad \forall p \in P$$

donde, dado $x \in (T \times P) \cup (P \times T)$, el valor de W_f se define como sigue:

$$W_f(x) = \begin{cases} 1 & \text{si } x \in F \\ 0 & \text{si } x \notin F \end{cases}$$

El disparo de una transición lo denotaremos por $M[t\rangle M'$.

□

Ejemplo 6 Consideremos el ejemplo 2.5. Si se dispara la transición t_1 el marcado de la red pasará a ser:

$$M'(p_1) = 0 \quad M'(p_2) = 0 \quad M'(p_3) = 1$$

□

Definición 17 (Activación concurrente de transiciones)
Sea $N = (P, T, F, M)$ una red de Petri ordinaria marcada, y $R \subseteq T$ un subconjunto del conjunto de transiciones de la red.

Se dice que las transiciones de R *están permitidas concurrentemente*, lo que denotaremos como $M[R\rangle$, si y sólo si:

$$\forall\, p \in P \quad M(p) \geq \sum_{t \in R} W_f(p,t)$$

$W_f(p,t)$ se define de igual forma a como lo hicimos en la definición anterior.

Esta definición puede ser extendida a multiconjuntos, permitiendo así que varias instancias de una misma transición sean disparadas en un solo paso. De esta forma, diremos que un multiconjunto de transiciones R está permitido bajo el marcado M si y sólo si:

$$\forall p \in P \quad M(P) \geq \sum_{t \in T} W_f(p,t) \times R(t)$$

El disparo de un multiconjunto de transiciones R bajo un marcado M conduce la red a un nuevo marcado M' dado por:

$$M'(p) = M(p) - \sum_{t \in T} (W_f(p,t) - W_f(t,p)).R(t)$$

Esta evolución de la red en un solo paso la denotaremos por $M[R\rangle M'$.

□

2.5. Redes de Petri con arcos temporizados

Las redes de Petri con arcos temporizados son una extensión temporizada de las Redes de Petri, donde los tokens están anotados con un valor real que indica el tiempo transcurrido desde su creación (su *edad*) y los arcos que conectan lugares

con transiciones tienen asociado un intervalo de tiempo, el cual limita la edad de los token consumidos para disparar la transición adyacente.

En el modelo particular que nosotros consideramos, algunas transiciones pueden ser urgentes, en el sentido de que una vez que estén habilitadas para su ejecución, no puede pasar el tiempo, y debe ser ejecutada alguna transición.

Definición 18 (Red de Petri con arcos temporizados etiquetada (LTAPN))
Sea *Act* un conjunto finito de acciones visibles. Definimos una *red de Petri con arcos temporizados etiquetada (LTAPN)* como una tupla:

$$N = (S,\, T,\, F,\, times,\, \lambda)$$

donde

- S es un conjunto finito de lugares.
- T es un conjunto finito de transiciones.
- $(S \cap T = \emptyset)$.
- F es la relación de flujo $(F \subseteq (S \times T) \cup (T \times S))$.
- *times* es una función que asocia un intervalo cerrado de tiempo a cada arco (s, t) en F, es decir:

$$times : F\,|_{S\times T} \longrightarrow\ \mathbb{R}_0^+ \times (\mathbb{R}_0^+ \cup \{\infty\})$$

- $\lambda : T \longrightarrow Act \cup \{\tau\}$ es una función que etiqueta cada transición con una acción visible o con la acción τ. Las transiciones etiquetadas con τ las denominaremos internas, y se consideran urgentes, ya que una vez habilitadas no puede pasar el tiempo, y éstas deben ser ejecutadas inmediatamente, a menos que entren en conflicto con alguna otra transición.

□

Como hemos comentado anteriormente, los tokens están anotados con valores reales, de modo que los marcados son definidos por medio de multiconjuntos de $\mathbb{R}_0^+$. Más concretamente, un marcado M es una función:

$$M : S \longrightarrow \mathcal{B}(\mathbb{R}_0^+)$$

donde $\mathcal{B}(\mathbb{R}_0^+)$ denota el conjunto de multiconjuntos finitos de números reales no negativos. Así, como es habitual, cada lugar es anotado con un cierto número de tokens, cada uno de los cuales tiene asociado un valor real no negativo (su edad).

Denotaremos el conjunto de marcados de N como $\mathcal{M}(N)$, y usando la notación clásica de conjuntos, denotaremos el número de tokens en un lugar s por $|M(s)|$.

Los únicos marcados iniciales que se permiten son aquellos M tales que $\forall s \in S$ y para todo $x > 0$ tenemos que $M(s)(x) = 0$, es decir, la edad inicial de un token es siempre 0.

Definición 19 (Red de Petri con arcos temporizados etiquetada y marcada)
Definimos una *red de Petri con arcos temporizados etiquetada y marcada* (en adelante nos referiremos a ellas como MLTAPN) como un par (N, M) donde N es una red de Petri con arcos temporizados etiquetada, y M es un marcado de la misma.

□

Definición 20 Lugares iniciales
Dada una MLTAPN (N, M) definimos el conjunto de *lugares iniciales* como:

$$Init(N) = \{s \in S \mid \ |M(s)| > 0\}$$

□

Como es habitual, desde el marcado inicial podemos obtener nuevos marcados, como consecuencia de la evolución de la red, bien por disparo de una transición, o por el paso del tiempo. Una red de Petri con arcos temporizados etiquetada con un marcado inicial arbitrario, puede ser representada gráficamente mediante la extensión de la representación de las redes P/T con la correspondiente información temporal. En particular podemos usar la edad de los tokens para representarlos. Por lo tanto, una MLTAPN tiene inicialmente una colección de valores cero etiquetando cada uno de los lugares.

Ejemplo 7 En la Figura 2.6 se muestra la MLTAPN que modeliza el problema del productor/consumidor. En ella, la transición t_1 representa la acción correspondiente al proceso de fabricación del productor, que requerirá entre 5 y 9 unidades de tiempo. La acción t_2 representa la acción de inclusión del objeto generado en el buffer. Por otro lado, la transición t_3 representa la acción de obtener un objeto del buffer,

y la transición t_4 modelará el procesamiento que realiza el consumidor del objeto extraído del buffer, cuya duración variará entre 4 y 6 unidades de tiempo. Los tokens iniciales del lugar p_5 representan la capacidad máxima del buffer, en este caso serán 3, mientras que los tokens en el lugar p_6 representan los objetos en el buffer que no han sido consumidos todavía. Para finalizar, podemos observar que el arco entre el lugar p_5 y la transición t_2 está etiquetado con $< 0, \infty >$, lo que indica que esos tokens podrán ser usados en cualquier instante en el futuro.

Observemos que si el tiempo para que se dispare una de las transiciones t_1 o t_4 expira, el sistema se bloqueará, ya que en los lugares p_1 ó p_4 tendremos tokens muertos.

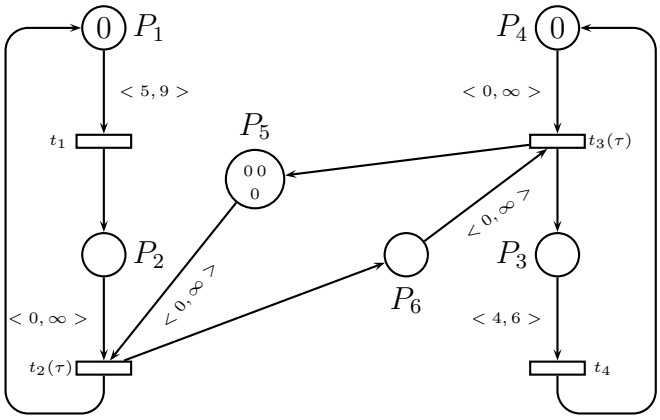


Figura 2.6: MLTAPN que modeliza el problema del productor/consumidor

□

Definición 21 (Regla de disparo)
Sea $N = (S, T, F, times, \lambda)$ una LTAPN, M un marcado de ésta y $t \in T$.

1. Decimos que t está habilitada en el marcado M si y sólo si:

$$\forall s \in {}^\bullet t \;\; \exists\, x_s \in \mathbb{R}_0^+ \;\; t.q. \;\; M(s)(x_s) > 0 \;\wedge\; x_s \in times(s,t)$$

   es decir, en cada precondición de t tenemos algún token cuya edad pertenece a $times(s,t)$. Como es habitual, ésto lo denotaremos como $M[t\rangle$.

2. Si t está habilitada en M, ésta podrá ser disparada, tras lo cual se alcanzará un nuevo marcado M' definido como sigue:

$$M'(s) = M(s) - C^-(s,t) + C^+(t,s), \forall s \in S$$

donde tanto la suma como la resta trabajan sobre multiconjuntos y

- $C^{-}(s,t) = \begin{cases} \{x_s\} & si\ s \in {}^{\bullet}t,\ x_s \in times(s,t),\ y\ M(s)(x_s) > 0 \\ \emptyset & \text{en otro caso} \end{cases}$
- $C^{+}(t,s) = \begin{cases} \emptyset & si\ s \notin t^{\bullet} \\ \{0\} & \text{en otro caso} \end{cases}$

Es decir, de cada lugar precondición de t se elimina un token que cumple la condición establecida en 1, y añadimos un nuevo token (con edad 0) en cada lugar postcondición de t. Denotamos esta evolución por $M[t\rangle M'$.

Debe notarse que esta evolución en general es no-determinista, porque cuando se dispara una transición, algunos de sus lugares precondición pueden tener varios tokens con edades diferentes que podrían ser usados para disparar la transición. Además, podemos ver que el disparo de una transición no consume tiempo. Así, para modelar el paso del tiempo necesitamos la función *age* que definimos a continuación. Cuando aplicamos esta función "envejecemos" todos los tokens de la red a la vez.

3. La función $age:\ M(N) \times \mathbb{R}_0^+ \longrightarrow \mathcal{M}(N)$ está definida como sigue:

$$age(M,x)(s)(y) = \begin{cases} M(s)(y-x) & si\ \ y \geq x \\ 0 & \text{en otro caso} \end{cases}$$

Una red marcada (N, M) permitirá el paso de x unidades de tiempo si y sólo si no existen transiciones etiquetadas con τ que se encuentren habilitadas bajo el marcado $age(M,y),\ \forall y \in [0,x]$.

En este caso, el marcado obtenido a partir de M después de x unidades de tiempo sin disparar ninguna transición vendrá dado por $age(M,x)$.

□

Aunque hemos definido la evolución mediante el disparo de una sola transición, ésta puede ser fácilmente extendida al disparo de pasos o de bolsas de transiciones. Las transiciones de una bolsa pueden ser disparadas juntas en un solo paso, o bien pueden ser disparadas secuencialmente en un determinado orden, ya que no se produce envejecimiento de los tokens con el disparo de las transiciones. De esta forma, obtenemos evoluciones mediante pasos que denotaremos por $M[R\rangle M'$.

Finalmente, mediante la alternancia entre transiciones de pasos y transiciones de paso de tiempo podemos definir una semántica de pasos temporizados, donde las secuencias de pasos temporizados son aquellas tales que:

$$\sigma = M_0 \, [R_1\rangle_{x_1} \, M_1 \, \ldots \, M_{n-1} \, [R_n\rangle_{x_n} \, M_n$$

donde M_i son marcados, R_i son multiconjuntos de transiciones y $x_i \in \mathbb{R}_0^+$ de modo que $M_i[R_{i+1}\rangle M'_{i+1}$ y $M_{i+1} = age(M'_{i+1}, x_{i+1})$.

Debemos hacer notar que se permite $x_i = 0$ para poder capturar la ejecución en tiempo cero de dos pasos relacionados causalmente.

Dada una MLTAPN (N, M_0), definimos $[M_0\rangle$ como el conjunto de los marcados alcanzables en la red N partiendo del marcado inicial M_0. Diremos que (N, M_0) es *limitado* si para todo $s \in S$ existe $n \in \mathbb{N}$ tal que para todo $M \in [M_0\rangle$ tenemos que $|M(s)| \leqslant n$.

Definición 22 (Token muerto)
Sea (N, M) una red marcada. Un token en un lugar s bajo un marcado M se dice que es *muerto* si nunca puede ser usado para disparar una transición, es decir, permanecerá en ese lugar para siempre, pero aumentando su edad indefinidamente.

Entonces diremos que un marcado es *muerto* si es nulo o todos sus tokens son *muertos*.

□

Definición 23 (Seguridad)
Decimos que (N, M) es *segura* si $|M(s)| \leqslant 1, \ \forall\, s \in P$ y $\forall\, M \in [M_0\rangle$. Diremos que un marcado es débilmente seguro si para cada marcado alcanzable tenemos a lo sumo un token no muerto en cada lugar.

□

Capítulo 3

Álgebra Temporizada (TPAL)

El modelo que presentamos, TPAL (Timed Process ALgebra), se basa en LOTOS, añadiéndole operadores nuevos para especificar el comportamiento temporal y las interacciones urgentes. Siguiendo las ideas de Quemada, Azcorra y de Frutos [QAdF89], introducimos un operador de prefijo temporizado, $a < t_1, t_2 >; P$, el cual establece un intervalo de tiempo para la ejecución de la acción correspondiente, y una vez agotado ese intervalo temporal, esa acción ya no puede ser ejecutada. Otros operadores que hemos incluido son el prefijo de acción urgente, $\tau; P$ y el operador de espera, $wait(t); P$. El primero establece el carácter de urgencia de la acción τ, en el sentido de que debe ser ejecutada inmediatamente, sin ningún retraso (a menos que tengamos un conflicto con otra acción interna), mientras que el segundo operador establece un retraso de t unidades de tiempo, y entonces se comporta como $\tau; P$. Además, mantenemos los operadores clásicos de LOTOS para la elección externa, composición paralela, ocultamiento y recursión.

Presentamos una semántica operacional para el lenguaje que captura el comportamiento anteriormente mencionado y particularmente el carácter de urgencia de las acciones internas.

3.1. Sintaxis

Sea Vis un conjunto finito de acciones, y $Act = Vis \cup \{\tau\}$, el conjunto de acciones incluida la acción τ.

La sintaxis del lenguaje TPAL (Timed Process ALgebra) está definida por la

siguiente expresión BNF:

$$N ::= stop \mid a < t_1, t_2 >; N \mid \tau; N \mid wait(t); N \mid N \,\Box\, N \mid N \parallel_A N \mid N \backslash a \mid X \mid \mu X.N$$

donde $A \subseteq Vis$, $X \in Idf$ (conjunto de identificadores), $a \in Vis$, $t, t_1 \in \mathbb{R}_0^+$ y $t_2 \in \mathbb{R}_0^+ \cup \{\infty\}$, siendo $\mathbb{R}_0^+ = \mathbb{R}^+ \cup \{0\}$.

Como es habitual denotaremos por Act^* al conjunto de secuencias de acciones de Act.

A continuación se hace una descripción informal e intuitiva de los operadores del lenguaje. Posteriormente, en la siguiente sección dotaremos al lenguaje de una semántica operacional, con lo cual quedará formalizado el significado de cada operador, así como la relación existente entre ellos.

stop .- Representa un deadlock, es decir, una máquina que no puede ejecutar ninguna acción, pero que permite el paso del tiempo.

Prefijo temporizado. $(a < t_1, t_2 >; N)$.- La acción a puede ser ejecutada dentro del intervalo $[t_1, t_2]$. Una vez que el intervalo ha expirado, se pierde la posibilidad de realizar esa acción, es decir, no se fuerza su ejecución cuando su time-out va a expirar. Una vez que la acción a se ha ejecutado, el proceso se comporta como N.

Prefijo de acción urgente. $(\tau; N)$.- La acción τ se ejecuta inmediatamente y el proceso pasa a comportarse como N.

Retraso. $(wait(t); N)$.- El proceso espera t unidades de tiempo, y después se comporta como $\tau; N$.

Elección externa. $(N_1 \,\Box\, N_2)$.- Representa el operador de elección externa clásico. Cuando se requiere que se ejecute una acción, ésta es ejecutada por el proceso que pueda realizarla, y si ambos pueden ejecutarla, entonces la elección del proceso es realizada de forma interna por el sistema.

Paralelo. $(N_1 \parallel_A N_2)$.- Representa la ejecución paralela de los procesos N_1 y N_2 sincronizando en las acciones del conjunto A.

Ocultamiento. $(N \backslash a)$.- Todas las instancias de la acción a en el proceso N son ocultadas y se convierten en urgentes (τ).

X.- Es un identificador, el cual en principio puede representar cualquier proceso del lenguaje.

Recursión. ($\mu X.N$).- Es el clásico operador de recursión, para la obtención de procesos infinitos.

En adelante, nos centraremos fundamentalmente en los términos cerrados y guardados que se derivan de esta sintaxis. Al conjunto de términos cerrados y guardados lo denotaremos como *TPAL*.

3.2. Semántica operacional

Con la definición de la semántica operacional, proporcionamos a los operadores del lenguaje un significado e interpretación precisa mediante la descripción de como un proceso puede transformándose en otro proceso. Esta evolución se presenta mediante un sistema de transiciones, el cual poseerá varios tipos de transiciones.

Concretamente, para el lenguaje TPAL tendremos dos tipos de transiciones, transiciones temporales y transiciones mediante acciones, las cuales pasamos a describir a continuación. Posteriormente, cuando el lenguaje sea ampliado con operadores probabilísticos, introduciremos un nuevo tipo de transiciones.

Definición 24 (Transición temporal)
Será representada de la forma siguiente:

$$N \underset{t}{\longmapsto} N'$$

y su significado intuitivo es que el proceso N pasa a comportarse como el proceso N' después de t unidades de tiempo.

□

Definición 25 (Transición de acción)
La representaremos de la forma siguiente:

$$N \xrightarrow{a} N'$$

Su significado es el habitual, es decir, el proceso N ejecuta la acción a y pasa a comportarse como N'.

□

Presentamos la semántica operacional de los términos cerrados y guardados de TPAL utilizando el estilo estructurado de Plotkin [Plo81] y Milner [Mil89].

En la tabla 3.1 podemos ver las reglas para este lenguaje, las cuales pasamos a explicar intuitivamente a continuación. En la descripción de las reglas representaremos por $N \xrightarrow{a}$ la posible evolución de un proceso N ejecutando la acción a.

La regla R1 indica que la única evolución posible del proceso *stop* es el paso del tiempo.

El conjunto de reglas R2 capturan la semántica del prefijo temporizado. Concretamente, la regla $R2a$ captura la evolución del proceso mediante la ejecución de una acción, mientras que la regla $R2b$ establece la evolución mediante el paso del tiempo y la regla $R2c$ establece que una vez que el tiempo para ejecutar una acción ha expirado, perdemos la posibilidad de ejecutar esta acción.

La regla R3 captura el carácter urgente de la acción interna τ.

El conjunto de reglas $R4$ capturan la semántica del operador wait, entre ellas, la regla $R4a$ establece la evolución por el paso del tiempo, mientras que la regla $R4b$ indica que una vez consumido todo el tiempo, la única evolución posible es mediante la ejecución de la acción interna τ.

Las reglas $R5a$ y $R5b$ establecen la semántica clásica de la elección externa, mientras que la regla $R5c$ establece que el tiempo en la elección externa puede pasar sólo cuando ambos componentes lo permiten.

El conjunto de reglas $R6$ son las clásicas para el operador paralelo, donde las reglas $R6a$ y $R6b$ posibilitan la evolución de los procesos de forma independiente, mediante la ejecución de acciones no pertenecientes al conjunto de sincronización, mientras que la regla $R6c$ establece la evolución de ambos procesos, mediante la realización de una acción de sincronización. Por último, la regla $R6d$ es similar a la $R5c$ y establece cuando está permitido el paso del tiempo dentro de un paralelo.

La semántica del operador de ocultamiento viene dada por el conjunto de reglas $R7$, donde la última de ellas indica que una vez que una acción ha sido ocultada, ésta se convierte en urgente. Como puede observarse, la regla $R7c$ introduce premisas negativas, sin embargo usando el mismo procedimiento que en [Gro93] puede probarse la consistencia del sistema de transiciones.

El conjunto de reglas $R8$ capturan la semántica clásica del operador de recursión.

Para finalizar este conjunto de reglas, tenemos la regla $R9$, la cual captura el carácter acumulativo del tiempo.

Definición 26 (Semántica Operacional de TPAL)
Definimos la semántica operacional de cualquier término cerrado y guardado de TPAL mediante el multiconjunto de transiciones que pueden ser inferidas aplicando el sistema de reglas definido en la tabla 3.1, donde cada transición aparece tantas veces como formas diferentes haya de derivarla.

□

Definición 27 (Computación $N \stackrel{<w>}{\Longrightarrow}_t N'$)
Sean $N,\ N',\ N'' \in TPAL$, tres procesos cerrados y guardados, $w \in (Act\ \cup\ \mathbb{R}_0^+)^*$ una secuencia temporizada (cadena de acciones y tiempos), $e \in Act$ y $t,\ t' \in \mathbb{R}_0^+$.

1. Sea $N \stackrel{e}{\longrightarrow} N'$ una posible evolución del proceso N. Entonces $N \stackrel{<e>}{\Longrightarrow}_0 N'$ es una computación.

2. Sea $N \underset{t}{\longmapsto} N'$ una posible evolución del proceso N. Entonces $N \stackrel{<t>}{\Longrightarrow}_t N'$ es una computación.

3. Sea $N \stackrel{<w>}{\Longrightarrow}_t N'$ una computación, y $N' \stackrel{e}{\longrightarrow} N''$ una posible evolución del proceso N'. Entonces $N \stackrel{<w.e>}{\Longrightarrow}_t N''$ es una computación.

4. Sea $N \stackrel{<w>}{\Longrightarrow}_t N'$ una computación, y $N' \underset{t'}{\longmapsto} N''$ una posible evolución de N'. Entonces $N \stackrel{<w.t'>}{\Longrightarrow}_{t+t'} N''$ es una computación.

□

Ejemplo 8 Consideremos de nuevo el sistema de cruce de un tren descrito en el ejemplo 3 del capítulo anterior de esta tesis.

Atendiendo a esa descripción tenemos que el sistema está compuesto por tres procesos, que representan respectivamente al tren, la barrera y el sistema de control. Estos procesos sincronizan en las acciones siguientes: *appr*, que representa el envío de la señal de aproximación; *exit*, que representa la señal que envía el tren cuando termina de cruzar; *bajar*, que representa la señal que envía el control a la barrera para que inicie la bajada; y *subir*, que es la acción de enviar la señal a la barrera para que inicie la subida.

La especificación de este sistema será la siguiente:

$$
\begin{aligned}
Tren = \quad & \mu X.aprox < 0,0 >; in < 1,\infty >; cruza < 4,4 >; exit < 0,0 >; X \\
Barrera = \quad & \mu Y.bajar < 0,\infty >; down < 1,3 >; subir < 0,\infty >; \\
& up < 1,2 >; Y \\
Control = \quad & \mu Z.aprox < 0,\infty >; bajar < 1,1 >; exit < 0,\infty >; \\
& subir < 1,1 >; Z \\
Sistema = \quad & ((Tren \parallel_{\{aprox,exit\}} Control) \parallel_{\{bajar,subir\}} Barrera) \\
& \backslash aprox \backslash bajar \backslash subir \backslash exit
\end{aligned}
$$

□

R1) $\text{stop} \underset{t}{\longmapsto} \text{stop} \quad \forall t \in \mathbb{R}_0^+$

R2a) $a < 0, t >; N \xrightarrow{a} N \quad \forall t \in \mathbb{R}_0^+$

R2b) $a < t_1, t_2 >; N \underset{t'}{\longmapsto} a < t_1 \dot{-} t', t_2 - t' >; N \;\; 0 \leq t' \leq t_2,$
donde $x \dot{-} y = \max\{0, x - y\}$

R2c) $a < 0, 0 >; N \underset{t'}{\longmapsto} \text{stop} \quad t' > 0$

R3) $\tau; N \xrightarrow{\tau} N$

R4a) $\text{wait}(t); N \underset{t'}{\longmapsto} \text{wait}(t - t'); N\,, \;\; 0 \leq t' \leq t$

R4b) $\text{wait}(0); N \xrightarrow{\tau} N$

R5a) $\dfrac{N_1 \xrightarrow{e} N_1'}{N_1 \Box N_2 \xrightarrow{e} N_1'}$ **R5b)** $\dfrac{N_2 \xrightarrow{e} N_2'}{N_1 \Box N_2 \xrightarrow{e} N_2'}$

R5c) $\dfrac{N_1 \underset{t'}{\longmapsto} N_1' \;\; N_2 \underset{t'}{\longmapsto} N_2'}{N_1 \Box N_2 \underset{t'}{\longmapsto} N_1' \Box N_2'}$

R6a) $\dfrac{N_1 \xrightarrow{e} N_1' \quad e \notin A}{N_1 \parallel_A N_2 \xrightarrow{e} N_1' \parallel_A N_2}$ **R6b)** $\dfrac{N_2 \xrightarrow{e} N_2' \quad e \notin A}{N_1 \parallel_A N_2 \xrightarrow{e} N_1 \parallel_A N_2'}$

R6c) $\dfrac{N_1 \xrightarrow{a} N_1', \quad N_2 \xrightarrow{a} N_2' \;\; a \in A}{N_1 \parallel_A N_2 \xrightarrow{a} N_1' \parallel_A N_2'}$ **R6d)** $\dfrac{N_1 \underset{t'}{\longmapsto} N_1', \quad N_2 \underset{t'}{\longmapsto} N_2'}{N_1 \parallel_A N_2 \underset{t'}{\longmapsto} N_1' \parallel_A N_2'}$

R7a) $\dfrac{N \xrightarrow{e} N' \;\; e \in Act \;\; e \neq a}{N \backslash a \xrightarrow{e} N' \backslash a}$ **R7b)** $\dfrac{N \xrightarrow{a} N'}{N \backslash a \xrightarrow{\tau} N' \backslash a}$

R7c) $\dfrac{N \underset{t}{\longmapsto} N', \neg N \xrightarrow{a} \wedge \neg[\exists\, t' < t,\; N \underset{t'}{\longmapsto} N''\,,\; N'' \xrightarrow{a}]}{N \backslash a \underset{t}{\longmapsto} N' \backslash a}$

R8a) $\dfrac{N\{\mu X.N/X\} \xrightarrow{e} N'}{\mu X.N \xrightarrow{e} N'}$ **R8b)** $\dfrac{N\{\mu X.N/X\} \underset{t}{\longmapsto} N'}{\mu X.N \underset{t}{\longmapsto} N'}$

R9) $\dfrac{N_1 \underset{t_1}{\longmapsto} N_2 \;\; N_2 \underset{t_2}{\longmapsto} N_3}{N_1 \underset{t_1+t_2}{\longmapsto} N_3}$

Tabla 3.1: Semántica operacional

Capítulo 4

Grafos de Estados Dinámicos

Los grafos de estados dinámicos son, esencialmente, autómatas extendidos con variables, que nosotros denominaremos relojes, las cuales nos permiten capturar información de tiempos locales para cada uno de los procesos. Algunos de estos relojes se actualizan cuando se ejecuta una transición, sincronizándolos con el instante de tiempo en que se ejecuta la transición. Esta es una de las diferencias con los autómatas temporizados [AD90, ACD90], en los cuales algunos de los relojes se resetean al realizar las transiciones.

Definición 28 (Grafo de estados dinámico)
Un *grafo de estados dinámico* es una tupla:

$$G = (V,\, E,\, \lambda,\, v_0,\, clocks)$$

donde:

- V es un conjunto de nodos.
- E es un conjunto de arcos.
- $\lambda : E \longrightarrow Act_\tau \times Times \times \mathcal{P}(clocks)$ es una función de etiquetado sobre los arcos, la cual asigna a cada arco una tupla compuesta por el nombre de una acción a, un conjunto de restricciones sobre los relojes y un conjunto de relojes que se sincronizan con el instante en el cual la acción es ejecutada.

  $Times : clocks \hookrightarrow \mathbb{R}_0^+ \times \mathbb{R}_0^+ \cup \{\infty\}$ es una función parcial que define las restricciones sobre los relojes, de modo que:

  $$Times(R_i) = (\alpha_i, \beta_i)$$

- v_0 es el nodo inicial del grafo.
- $clocks = \{R_1,\ R_2,\ \ldots,\ R_n\}$, es el conjunto de relojes asociado al grafo. Entre estos relojes existe uno especial, que denotaremos por R_g, el cual mantendrá la información sobre el tiempo global del sistema.

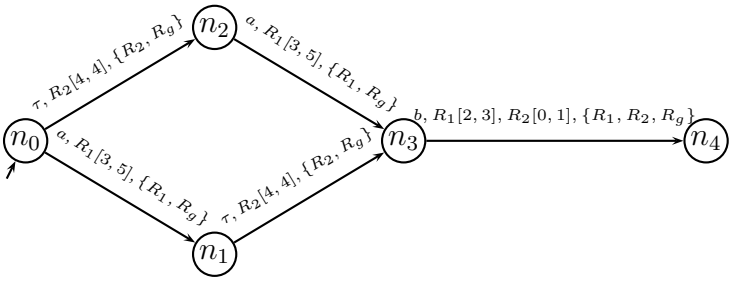
□

Los grafos de estados dinámicos, al igual que ocurre con los autómatas temporizados, se representan gráficamente mediante un grafo dirigido, donde los arcos están etiquetados con el valor que les asigna la función de etiquetado λ.

Ejemplo 9 Consideremos el grafo definido como sigue:

$$
\begin{array}{ll}
V = & \{n_0, n_1, n_2, n_3, n_4\} \\
E = & \{(n_0, n_1), (n_0, n_2), (n_1, n_3), (n_2, n_3), (n_3, n_4)\} \\
v_0 = & n_0 \\
\lambda(n_0, n_1) = & (a, R_1[3, 5], \{R_1, R_g\}) \\
\lambda(n_0, n_2) = & (\tau, R_2[4, 4], \{R_2, R_g\}) \\
\lambda(n_1, n_3) = & (\tau, R_2[4, 4], \{R_2, R_g\}) \\
\lambda(n_2, n_3) = & (a, R_1[3, 5], \{R_1, R_g\}) \\
\lambda(n_3, n_4) = & (b, R_1[2, 3], R_2[0, 1], \{R_1, R_2, R_g\}) \\
clocks = & \{R_1, R_2, R_g\}
\end{array}
$$

La representación gráfica del mismo puede verse en la Figura 4.1.



$clocks = \{R_1, R_2, R_g\}$

Figura 4.1: Grafo de estados dinámico

□

4.1. Semántica

La semántica de los grafos de estados dinámicos vendrá dada por las reglas que establecen las posibles evoluciones de éstos. Antes de definir estas reglas introduciremos unas definiciones previas.

Definición 29 (Estado)
Dado un grafo $G = (V, E, \lambda, v_0, \mathit{clocks})$, definimos un *estado* del mismo como una terna (n, C, t), donde:

- $n \in V$ es un nodo
- C: $\mathit{clocks} \longrightarrow \mathbb{R}_0^+$ es una función que indica el valor de cada uno de los relojes en este estado.
- t es el instante actual. Este valor deberá cumplir la restricción $t \geq C(R_g)$.

□

Para cualquier grafo G, su estado inicial será $s_0 = (n_0, C_0, 0)$ donde $C_0(R_j) = 0$, $\forall\, R_j \in \mathit{clocks}$.

Definición 30 (Espera mínima)
Sea $G = (V, E, \lambda, v_0, \mathit{clocks})$ un grafo de estados dinámico, el cual se encuentra en un estado $s = (n, C, t)$ y consideremos un arco e cuyo nodo origen es el nodo n, y cuya etiqueta es $\lambda(e) = (a, r, S)$.

La *espera mínima* en el nodo n para unos valores de los relojes C y un determinado arco e, representada como $\mathit{Min}(n, C, e)$, es la cantidad mínima de tiempo que debe transcurrir desde la llegada al nodo n hasta que la transición asociada a e puede ser ejecutada.

Este valor será calculado como sigue:

$$\begin{aligned}\mathit{Min}(n, C, e) = \quad & \text{máx}\{C(R_j) + \alpha_j \mathbin{\dot{-}} C(R_g) \mid R_j \in \mathit{clocks}, \\ & \qquad r(R_j) \text{ está definida y } r(R_j) = [\alpha_j, \beta_j]\}\end{aligned}$$

donde $a \mathbin{\dot{-}} b = \text{máx}(0, a - b)$.

Esta espera mínima viene definida a partir de las restricciones sobre los relojes que hay establecidas para el arco e, así como por el valor de los relojes C.

□

Ejemplo 10 Volvamos sobre el grafo del ejemplo 9 (Figura 4.1). Consideremos que el estado en el que se encuentra el mismo es $s_3 = (n_3, C_3, 4)$, donde $C_3(R_1) = 3$, $C_3(R_2) = 4$ y $C_3(R_g) = 4$.

El valor de $Min(n, C, e)$ para el único arco que parte del nodo n_3 será:

$$Min(n_3, C_3, e) = \text{máx}(3 + 2 - 4, 4 + 0 - 4) = 1$$

□

Definición 31 (Espera máxima)
Sea $G = (V, E, \lambda, v_0, clocks)$ un grafo de estados dinámico, $s = (n, C, t)$ un determinado estado del mismo y e un arco cuyo nodo origen es el nodo n, y cuya etiqueta es $\lambda(e) = (a, r, S)$.

La *espera máxima*, en el nodo n para unos valores de los relojes C y un determinado arco e, denotada como $Max(n, C, e)$, es la cantidad de tiempo que puede transcurrir en el nodo n de modo que la transición asociada a e pueda ser ejecutada, de acuerdo con las restricciones r establecidas en ese arco.

El valor de esta espera máxima se calcula como sigue:

$$\begin{aligned} Max(n, C, e) = \quad & \text{mín}\{C(R_j) + \beta_j - C(R_g) \mid R_j \in clocks, \\ & r(R_j) \text{ está definida y } r(R_j) = [\alpha_j, \beta_j]\} \end{aligned}$$

□

Ejemplo 11 Consideremos el grafo del ejemplo 9, y supongamos que éste se encuentra en el estado $(n_3, C_3, 4)$, donde $C_3(R_1) = 3$, $C_3(R_2) = 4$ y $C_3(R_g) = 4$. Calculamos $Max(n, C, e)$ para el único arco que parte del nodo $n3$ y obtendremos:

$$Max(n_3, C_3, e) = \text{mín}(3 + 3 - 4, 4 + 1 - 4) = 1$$

□

Definición 32 ($Min_\tau(n, C)$)
Sea el grafo $G = (V, E, \lambda, v_0, clocks)$, el cual se encuentra en un estado $s = (n, C, t)$.

Dados unos valores C de los relojes del grafo, $Min_\tau(n, C)$ es la cantidad máxima de tiempo que puede transcurrir en el nodo n desde que éste fue alcanzado. Esta cantidad de tiempo viene determinada por las transiciones etiquetadas con τ que parten del nodo n.

Una vez agotado ese tiempo, no podrá pasar más tiempo y habrá que ejecutar alguna transición, ya sea una etiquetada con τ o cualquier otra permitida.

Este valor puede calcularse de la forma siguiente:

$$Min_\tau(n,C) = \begin{cases} \infty & \text{si } \nexists\, e \text{ t.q. } n \xrightarrow{e},\ \lambda_1(e) = \tau \\ \text{mín}\{Min(n,C,e) \mid n \xrightarrow{e},\ \lambda_1(e) = \tau\} & \text{en caso contrario} \end{cases}$$

donde $n \xrightarrow{e}$ indica que existe un arco etiquetado con e que sale del nodo n.

□

Ejemplo 12 Volvamos a centrar nuestra atención en el grafo del ejemplo 9, y consideramos que éste se encuentra en el estado $(n_3, C_3, 4)$, donde $C_3(R_1) = 3$, $C_3(R_2) = 4$ y $C_3(R_g) = 4$. Como no existe ningún arco etiquetado con τ que parta del nodo n_3 tendremos que $Min_\tau(n_3, C_3) = \infty$.

Consideremos ahora que el grafo se encuentra en el estado inicial $(n_0, C_0, 0)$. En este caso existe un arco etiquetado con τ por lo que aplicando la definición anterior obtendremos que $Min_\tau(n_0, C_0) = 4$.

□

Definición 33 (Condición de habilitación)
Sea $G = (V, E, \lambda, v_0, clocks)$ un grafo de estados dinámicos, el cual se encuentra en un determinado estado $s = (n, C, t)$, y sea e un arco cuyo nodo origen es el nodo n.

Diremos que e *está habilitado* para ser ejecutado si y sólo si se cumple la condición:

$$Min(n,C,e) \leq \Delta t \leq Max(n,C,e) \wedge \Delta t \leq Min_\tau(n,C)$$

donde $\Delta t = t - C(R_g)$.

Esta condición será representada como $act(s, e)$.

□

Llegados a este punto estamos en condiciones de establecer las reglas que definen la semántica de los grafos, las cuales se muestran en la Tabla 4.1.

La regla $G1$ indica como se producirá la evolución del grafo mediante la ejecución de una acción y los cambios que provocará sobre el estado del grafo. Concretamente se cambiará de nodo y se actualizarán los valores de todos los relojes sincronizados, asignándolos al instante t.

G1) $$\frac{s = (n, C, t), \quad act(s, e)}{s \xrightarrow{e} (m, adjust(C, e, t), t)}$$

G2) $$\frac{s = (n, C, t), \quad t' \geq t, \quad t' - C(R_g) \leq Min_\tau(n, C)}{s \underset{t'-t}{\longmapsto} (n, C, t')}$$

Tabla 4.1: Reglas de transición para los grafos

La regla $G2$ establece la evolución mediante el paso del tiempo. En este caso no se modificará ni el nodo ni los valores de los relojes, siendo modificado solamente el valor de t.

En la definición de las reglas, s es un estado (n, C, t), e representa una transición del grafo entre los nodos n y m, y la función *adjust*, que actualiza los valores de los relojes, está definida como sigue:

$$adjust(C, e, t)(R_i) = \begin{cases} t & \text{si } R_i \in \lambda_3(e) \\ C(R_i) & \text{en caso contrario} \end{cases}$$

Denotaremos por $Reach(G, s)$ al conjunto de estados alcanzables en G desde el estado s, de acuerdo con estas reglas.

Definición 34 (Computación $s_0 \overset{<w>}{\Longrightarrow}_t s$)
Sea $G = (V, E, \lambda, n_0, clocks)$ un grafo de estados dinámico, $w \in (Act \cup \mathbb{R}_0^+)^*$ una secuencia temporizada (cadena de acciones y tiempos), $e \in Act$ y $t, t' \in \mathbb{R}_0^+$.

1. Sea $s_0 = (n_0, C_0, 0)$ el estado inicial de G, y $s_0 \xrightarrow{e} s'$ una posible evolución de G. Entonces $s_0 \overset{<e>}{\Longrightarrow}_0 s'$ es una computación.

2. Sea $s_0 = (n_0, C_0, 0)$ el estado inicial de G y $s_0 \underset{t}{\longmapsto} s'$ una posible evolución de G. Entonces $s_0 \overset{<t>}{\Longrightarrow}_t s'$ es una computación.

3. Sea $s_0 \overset{<w>}{\Longrightarrow}_t s'$ una computación en G, y $s' \xrightarrow{e} s''$ una posible evolución de G. Entonces $s_0 \overset{<w.e>}{\Longrightarrow}_t s''$ es una computación.

4. Sea $s_0 \overset{<w>}{\Longrightarrow}_t s'$ una computación en G y $s' \underset{t'}{\longmapsto} s''$ una posible evolución de G. Entonces $s_0 \overset{<w.t>}{\Longrightarrow}_{t+t'} s''$ es una computación.

□

El lema siguiente establece que, dado un nodo del grafo n, sea cual sea el instante de tiempo de partida, si todos los relojes se encuentran sincronizados con ese instante, las posibles evoluciones del grafo serán las mismas, aunque éstas estarán desplazadas en el tiempo.

Lema 1 Dado un grafo $G = (N, E, \lambda, n_0, clocks)$, su estado inicial $s_0 = (n_0, C_0, 0)$ y un estado $s'_0 = (n_0, C'_0, t')$, con $t' \in \mathbb{R}^+$ y $C'_0(R_j) = t',\ \forall R_j \in clocks$.

Entonces:

$$(n, C, t) \in Reach(G, s_0) \iff (n, C', t + t') \in Reach(G, s'_0)$$

donde $C'(R_j) = C(R_j) + t',\ \forall R_j \in clocks$

Demostración: En esta demostración procederemos por inducción sobre el número de reglas $G1$ ó $G2$ aplicadas para obtener el estado (n, C, t).

- Caso base (número de reglas=1)
  Para este caso base distinguiremos dos posibles casos, dependiendo de la regla aplicada.

  - Caso b.1
    La regla aplicada es $G1$ por lo que el nuevo estado (n, C, t) es alcanzado mediante la ejecución de una transición e, es decir $(n_0, C_0, 0) \xrightarrow{e} (n, C, t)$.
    Probemos en primer lugar que la transición e puede ejecutarse en el estado $(n_0, C_0, 0)$ si y sólo si puede ejecutarse en el estado (n_0, C'_0, t').
    La transición e puede ejecutarse si y sólo si la condición de habilitación para esa transición se cumple en el estado inicial, es decir, si se cumple:

    $$(n, C, t) \in Reach(G, s_0)$$

    y ésto ocurre si y sólo si:

    $$Min(n_0, C_0, e) \leq t - C_0(R_g) \leq Max(n_0, C_0, e) \wedge t - C_0(R_g) \leq Min_\tau(n_0, C_0)$$

    Tanto en el estado $(n, C_0, 0)$ como en el estado (n, C'_0, t') nos encontramos en el mismo nodo siendo el valor de todos los relojes el mismo, por lo que se cumplirán las igualdades siguientes: $Min_\tau(n_0, C_0) = Min_\tau(n_0, C'_0)$, $Min(n_0, C_0, e) = Min(n_0, C'_0, e)$, $Max(n_0, C_0, e) = Max(n_0, C'_0, e)$ y $C_0(R_j) = C'_0(R_j) - t'$.

Aplicando estas igualdades en la condición de habilitación tendremos:

$$Min(n_0, C'_0, e) \leq t + t' - C'_0(R_g) \leq Max(n_0, C'_0, e) \wedge$$
$$t + t' - C'_0(R_g) \leq Min_\tau(n_0, C'_0)$$

que será cierto si y sólo si en el estado (n_0, C'_0, t') podemos ejecutar la transición e y llegar al estado $(n, C', t + t')$, es decir, si y sólo si:

$$(n, C', t + t') \in Reach(G, s'_0)$$

Probemos ahora que los valores de los relojes de ambos estados cumplen $C'(R_j) = C(R_j) + t'$. Para ello distinguiremos dos casos dependiendo de si el reloj pertenece al conjunto S de relojes que deben sincronizarse con la ejecución de la transición e.

- Caso b.1.1 ($R_j \notin S$)
  Su valor no se actualiza, por lo que se cumplirá $C(R_j) = C_0(R_j)$ y $C'(r_j) = C'_0(R_j)$. Como se cumple que $C'_0(R_j) = C_0(R_j) + t'$ se cumplirá que $C'(R_j) = C(R_j) + t'$.
- Caso b.1.2 ($R_j \in S$)
  En este caso el valor de cada uno de los relojes se actualizará al instante en que se ejecuta la acción, de modo que su nuevo valor en el estado (n, C, t) será 0 y en el estado (n, C', t') será t', es decir sus valores no se verán modificados. Por tanto se cumplira que $C'(R_j) = C(R_j) + t'$.

- Caso b.2
  La regla aplicada es la $G2$, por lo que la transición es de paso de tiempo. Esto hace que permanezcamos en el mismo nodo, con los mismos valores para los relojes y la única variación es el valor de t, por lo que el estado alcanzado será (n_0, C_0, t).

  Así $(n_0, C_0, t) \in Reach(G, s_0)$ si y sólo si se cumple que $t - C_0(R_g) \leq Min_\tau(n_0, C_0)$.

  Por los valores de los relojes tenemos que $Min_\tau(n_0, C_0) = Min_\tau(n_0, C'_0)$, cumpliéndose además que $C_0(R_j) = C'_0(R_j) - t'$.

  Si aplicamos estas igualdades a las formula anterior obtenemos que $t + t' - C'_0(R_g) \leq Min_\tau(n_0, C'_0)$, lo que se cumplirá si y sólo si en el estado (n_0, C'_0, t') se permite el paso de $t + t'$ unidades de tiempo, y por tanto $(n_0, C'_0, t + t') \in Reach(G, s'_0)$.

- Caso general (número de reglas>1)
  Consideramos que la secuencia de reglas aplicadas es $G = G_n \cdot Gi\,(i = 1, 2)$, donde G_n representa una secuencia de n aplicaciones de las reglas $G1$ ó $G2$. Si aplicamos la hipótesis de inducción sobre la secuencia G_n, tendremos que:

  $$(n_1, C_1, t_1) \in \mathit{Reach}(G, s_0) \iff (n_1, C_1', t_1 + t') \in \mathit{Reach}(n_0, s_0')$$

  con $C_1'(R_j) = C_1(R_j) + t'$ siendo (n_1, C_1, t_1) el estado obtenido tras G_n.

  A partir del estado (n_1, C_1, t_1) aplicando la regla $G1$ ó $G2$ alcanzaremos un estado $(n, C, t) \in \mathit{Reach}(G, s_0)$. Al igual que en el caso base distinguiremos dos casos dependiendo de la regla utilizada para alcanzar el estado (n, C, t).

  - Caso g.1
    La regla aplicada es $G1$ por lo que el nuevo estado (n, C, t) es alcanzado mediante la ejecución de una transición e, es decir $(n_1, C_1, t_1) \xrightarrow{e} (n, C, t)$.

    Probemos en primer lugar que la transición e puede ejecutarse en el estado $s_1 = (n_1, C_1, t_1)$ si y sólo si puede ejecutarse en el estado $s_1' = (n_1, C_1', t_1 + t')$.

    La transición e puede ejecutarse si y sólo si la condición de habilitación para esa transición se cumple en el estado de partida, es decir, si se cumple:

    $$(n, C, t) \in \mathit{Reach}(G, s_1)$$

    y ésto ocurre si y sólo si:

    $$\mathit{Min}(n_1, C_1, e) \leq t - C_1(R_g) \leq \mathit{Max}(n_1, C_1, e) \wedge t - C_1(R_g) \leq \mathit{Min}_\tau(n_1, C_1)$$

    Tanto en el estado (n_1, C_1, t_1) como en el estado $(n_1, C_1', t_1 + t')$ nos encontramos en el mismo nodo siendo el valor de todos los relojes el mismo, por lo que se cumplirán las igualdades siguientes: $\mathit{Min}_\tau(n_1, C_1) = \mathit{Min}_\tau(n_1, C_1')$, $\mathit{Min}(n_1, C_1, e) = \mathit{Min}(n_1, C_1', e)$, $\mathit{Max}(n_1, C_1, e) = \mathit{Max}(n_1, C_1', e)$ y $C_1(R_j) = C_1'(R_j) - t'$.

    Aplicando estas igualdades en la condición de habilitación tendremos que

    $$\mathit{Min}(n_1, C_1', e) \leq t + t' - C_1'(R_g) \leq \mathit{Max}(n_1, C_1', e) \wedge$$
    $$t + t' - C_0'(R_g) \leq \mathit{Min}_\tau(n_1, C_1')$$

    que será cierto si y sólo si en el estado (n_1, C_1', t') podemos ejecutar la transición e y llegar al estado $(n, C', t + t')$, es decir, si y sólo si:

    $$(n, C', t + t') \in \mathit{Reach}(G, s_1')$$

Probemos ahora que los valores de los relojes de ambos estados cumplen $C'(R_j) = C(R_j) + t'$. Para ello distinguiremos dos casos dependiendo de si el reloj pertenece al conjunto S de relojes que deben sincronizarse con la ejecución de la transición e.

- Caso g.1.1 ($R_j \notin S$)
  Su valor no se actualiza, por lo que se cumplirá $C(R_j) = C_1(R_j)$ y $C'(r_j) = C'_1(R_j)$. Como se cumple que $C'_1(R_j) = C_1(R_j) + t'$ se cumplirá que $C'(R_j) = C(R_j) + t'$.
- Caso b.1.2 ($R_j \in S$)
  En este caso el valor de cada uno de los relojes se actualizará al instante en que se ejecuta la acción de modo que su nuevo valor en el estado (n, C, t) será t_1 y en el estado $(n, C', t_1 + t')$ será $t_1 + t'$, es decir sus valores no se verán modificados. Por tanto se cumplira que $C'(R_j) = C(R_j) + t'$.

- Caso g.2
  Podemos repetir el mismo razonamiento que en el caso b.2 y obtendremos que $(n, C', t + t')$ es alcanzable desde el estado $(n_1, C'_1, t_1 + t')$. si y sólo si (n, C, t) es alcanzable a partir del estado (n, C, t) con $C'(R_j) = C(R_j) + t'$ Por tanto se cumplirá $(n, C', t + t') \in Reach(G, s'_0)$ con $C'(R_j) = C(R_j) + t'$.

□

4.2. Traducción de TPAL a grafos

En esta sección procederemos a definir de forma estructural el grafo asociado a los términos regulares del álgebra TPAL, los cuales denotaremos por *RTPAL*. Estos términos son aquellos términos cerrados para los cuales, los procesos de la forma $\mu X.P$ no tienen instancias del operador paralelo y del operador ocultamiento en P y además, todas las ocurrencias de X están guardadas.

La última restricción (X guardadas) ya había sido impuesta en la definición de la semántica operacional, para dar una única interpretación a los términos recursivos. Las nuevas restricciones, concernientes a la prohibición de apariciones del operador paralelo y del operador de ocultamiento dentro de la recursión evitará que los grafos

obtenidos por medio de esta traducción sean infinitos (caso del paralelo), y harán que nuestra definición sea correcta (caso del ocultamiento).

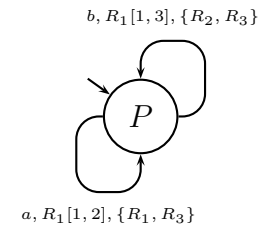
En el ejemplo siguiente podremos comprobar lo que ocurre si no se cumplen estas restricciones.

Ejemplo 13 Consideremos el proceso siguiente:

$$P = \mu.X(a < 1, 2 >; X \parallel b < 1, 3 >; X)$$

Para obtener su grafo deberíamos desplegar la recursión que aparece en ambos términos del paralelo, lo que nos volvería a introducir un nuevo paralelo y así sucesivamente, de modo que al construir el grafo obtendríamos un grafo infinito.

Podemos pensar que para este proceso tan simple podríamos obtener un grafo finito como el mostrado en la Figura 4.2. Sin embargo, este grafo no tiene el mismo comportamiento que el proceso P, lo cual puede comprobarse fácilmente.



$$clocks = \{R_1,\ R_2,\ R_3\}$$

Figura 4.2: Grafo de estados dinámico

□

Por simplicidad y claridad de la definición, nos referiremos a los nodos con el término que representan, lo que hará que podamos tener varios nodos etiquetados con el mismo término. No obstante, eso no supondrá ningún problema para comprender la definición.

4.2.1. Stop

El grafo del proceso *stop* consta de un único nodo etiquetado como *stop*, y poseerá un único reloj, que será el reloj global del sistema. Formalmente :

$$G[\![stop]\!] = (\{stop\}, \emptyset, \emptyset, stop, \{R_g\})$$

.

4.2.2. Prefijo temporizado

El grafo para el proceso $P = a < t_1, t_2 >; P_1$ será obtenido a partir del grafo $G[\![P_1]\!]$ al que se le añadirá un nuevo nodo y un nuevo arco. El conjunto de relojes, así como el reloj global serán heredados del grafo de P_1.

Para formalizar esta definición, consideremos el grafo asociado al proceso P_1, $G[\![P_1]\!] = (N_1, E_1, \lambda_1, n_{01}, clocks_1)$.

El grafo asociado al proceso P será:

$$G[\![P]\!] = (N, E, \lambda, P, clocks_1)$$

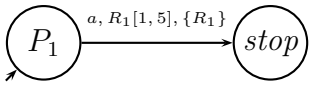
donde:

$$\begin{aligned} N &= N_1 \cup \{P\} \\ E &= E_1 \cup \{P \xrightarrow{e} n_{01}\} \\ \lambda(\bar{e}) &= \begin{cases} \lambda_1(\bar{e}) & \forall \bar{e} \in E_1 \\ (a, Intg, clocks_1) & \text{si } \bar{e} = e \end{cases} \end{aligned}$$

Siendo *Intg* el conjunto de restricciones definidas para esa transición. En este caso, la restricción se definirá solamente sobre el reloj global, siendo $Intg(R_g) = [t_1, t_2]$.

El tamaño de $G[\![P]\!]$ será el tamaño de $G[\![P_1]\!]$ incrementado en un nodo y un arco.

Ejemplo 14 Consideremos el proceso $P_1 = a < 1, 5 >; stop$. El grafo asociado será el mostrado en la Figura 4.3.



$clocks = \{R_1\}$

Figura 4.3: Grafo de P_1

□

4.2.3. Prefijo de acción urgente

La construcción del grafo asociado al proceso $P = \tau; P_1$ es similar a la dada en el apartado anterior para el prefijo temporizado, diferenciándose de aquélla en

la función de etiquetado, y concretamente en la definición de la etiqueta que se le asigna al nuevo arco introducido. La función de etiquetado en este caso será:

$$\lambda(\bar{e}) = \begin{cases} \lambda_1(\bar{e}) & \forall \bar{e} \in E_1 \\ (\tau, Intg, clocks_1) & \text{si } \bar{e} = e \end{cases}$$

donde $Intg(R_g) = [0, 0]$.

Ejemplo 15 Consideremos el proceso $P_2 = \tau; P_1$, donde P_1 es el proceso definido en el ejemplo 14, cuyo grafo se muestra en la Figura 4.3. El grafo de P_2 será el mostrado en la Figura 4.4.

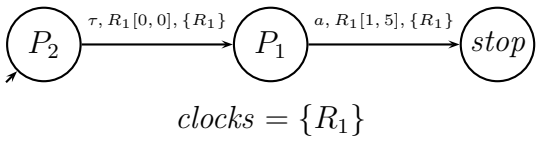


Figura 4.4: Grafo de P_2

□

4.2.4. Operador wait

Para el proceso $P = wait(t); P_1$, al igual que para el operador de prefijo de acción urgente, la construcción del grafo es similar a la del operador prefijo temporizado, variando solamente la función de etiquetado y concretamente la restricción establecida. Así, la función de etiquetado será:

$$\lambda(\bar{e}) = \begin{cases} \lambda_1(\bar{e}) & \forall\, \bar{e} \in E_1 \\ (\tau, Intg, clocks_1) & \text{si } \bar{e} = e \end{cases}$$

con $Intg(R_g) = [t, t]$.

Ejemplo 16 Consideremos el proceso $P_3 = wait(7); P_1$, donde P_1 es el proceso del ejemplo 14, cuyo grafo se muestra en la Figura 4.3. El grafo de P_3 será el mostrado en la Figura 4.5.

□

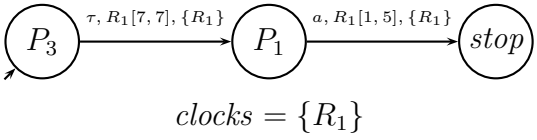


Figura 4.5: Grafo de P_3

4.2.5. Elección externa

Para construir el grafo de un proceso de la forma $P = P_1 \Box P_2$ debemos establecer la restricción de que el nodo inicial de los grafos $G[\![P_1]\!]$ y $G[\![P_2]\!]$ no debe tener arcos entrantes. De no imponerse esta restricción, el comportamiento que tendrá el grafo obtenido será completamente diferente al comportamiento establecido por la semántica operacional para el proceso $P = P_1 \Box P_2$.

En el siguiente ejemplo podemos observar lo que ocurriría si esta restricción no se cumple.

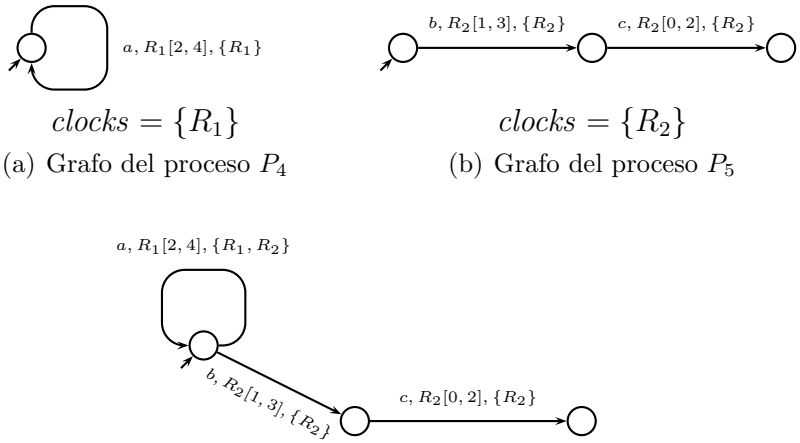
Ejemplo 17 Consideremos los procesos $P_4 = \mu X.a < 2, 4 >; X$ y $P_5 = b < 1, 3 > ; c < 0, 2 >; stop$.

Para construir el grafo del proceso P_4, aplicamos la definición para la recursión con lo que obtendremos un grafo con un único nodo que será el inicial (ver Figura 4.6(a)). Para el proceso P_5 aplicando las construcciones correspondientes obtendremos el grafo de la Figura 4.6(b). Para construir el grafo del proceso $P_6 = P_4 \,\Box\, P_5$ la idea en principio es unir los grafos de ambos procesos eliminando ambos nodos iniciales y creando un nuevo nodo que será el inicial del nuevo grafo, de modo que todos los arcos que salían de ambos nodos iniciales ahora salen del nuevo, y todos los que llegaban a esos nodos ahora llegan al nuevo nodo. De esta forma se obtendría el grafo mostrado en la Figura 4.6(c).

Como puede observarse $G[\![P_6]\!]$ tras ejecutar la acción a nos permite volver a elegir entre la acción a o la acción b, mientras que en el proceso P_6, y de acuerdo con la semántica del mismo, tras realizar la acción a solamente podríamos seguir realizando la acción a.

□

Para conseguir que los nodos iniciales de los grafos no posean arcos entrantes, lo que haremos será desplegar la recursión que pueda aparecer en ambas componentes al primer nivel.



$clocks = \{R_1\}$

(a) Grafo del proceso P_4

$clocks = \{R_2\}$

(b) Grafo del proceso P_5

$clocks = \{R_1, R_2\},\ R_g = R_2$

(c) Grafo del proceso $P_6 = P_4 \Box P_5$

Figura 4.6: Grafo para el proceso del ejemplo 17

Definición 35 (Unfold)
Definiremos esta operación, la cual actúa a nivel sintáctico sobre los términos del álgebra, modificándolos con el fin de garantizar el cumplimiento de la restricción impuesta. El término obtenido es equivalente en conducta al inicial, pues únicamente se realiza el despliegue de la recursión, lo cual es compatible con nuestra semántica operacional.

$$
\begin{aligned}
&Unfold(stop) = stop \\
&Unfold(a < t_1, t_2 >; P) = a < t_1, t_2 >; P \\
&Unfold(\tau; P) = \tau; P \\
&Unfold(wait(t); P) = wait(t); P \\
&Unfold(P_1 \,\Box\, P_2) = Unfold(P_1) \,\Box\, Unfold(P_2) \\
&Unfold(P_1 \parallel_A P_2) = Unfold(P_1) \parallel_A Unfold(P_2) \\
&Unfold(P \backslash a) = Unfold(P) \backslash a \\
&Unfold(\mu X.P) = Unfold(P\{\mu X.P/X\})
\end{aligned}
$$

□

Podemos observar que esta definición es correcta, porque se están considerando términos regulares, y más concretamente, para un proceso $\mu X.P$, todas las apariciones de un identificador X en P deben estar prefijadas, lo que garantiza que $Unfold(\mu X.P)$ está bien definida.

Una vez definida esta operación, podemos construir el grafo asociado al proceso $P = P_1 \Box P_2$ de la forma siguiente:

$$G[\![P]\!] = G[\![\mathit{Unfold}(P_1)]\!] \,\Box_G\, G[\![\mathit{Unfold}(P_2)]\!]$$

El operador $\Box_G$ construye un nuevo grafo a partir de dos grafos que cumplen las restricciones establecidas anteriormente.

El conjunto de nodos del nuevo grafo tendrá todos los nodos de ambos grafos a excepción de los iniciales, los cuales serán sustituidos por un nuevo nodo, que será el nodo inicial del nuevo grafo. El conjunto de arcos estará formado por los arcos de ambos grafos menos aquéllos cuyo nodo origen fuera alguno de los nodos eliminados. Estos arcos serán sustituidos por nuevos arcos, cuyo nodo origen será el nuevo nodo que ha sido introducido, y cuyo nodo destino es el mismo que tenía el arco al que sustituye.

El reloj global del nuevo grafo será uno de los relojes globales de los grafos de partida, debiendo ser incluido éste en las etiquetas de todos los arcos del grafo.

La definición formal del operador $\Box_G$ se muestra a continuación.

Consideremos dos grafos $G_i = (N_i, E_i, \lambda_i, n_{0i}, \mathit{clocks}_i)$, $i = 1, 2$, cuyos estados iniciales no tienen arcos entrantes, definimos:

$$G_1 \,\Box_G\, G_2 = (N, E, \lambda, n_0, \mathit{clocks}_1 \cup \mathit{clocks}_2)$$

siendo:

$$\begin{array}{ll}
N = & (N_1 \cup N_2 \cup \{P\}) \setminus \{n_{01}, n_{02}\} \\
n_0 = & P \\
E = & ((E_1 \cup E_2) \setminus \\
 & (\{(n_{01}, n_1) \,|\, (n_{01}, n_1) \in E_1\} \cup \{(n_{02}, n_2) \,|\, (n_{02}, n_2) \in E_2\})) \cup \\
 & (\{(n_0, n_1) \,|\, (n_{01}, n_1) \in E_1\} \cup \{(n_0, n_2) \,|\, (n_{02}, n_2) \in E_2\}) \\
R_g = & R_{g1} \text{ ó } R_{g2} \\
\lambda(e) = & (\bar{\lambda}_1(e),\ \bar{\lambda}_2(e),\ \bar{\lambda}_3(e) \cup \{R_g\}) \quad \forall e \in E
\end{array}$$

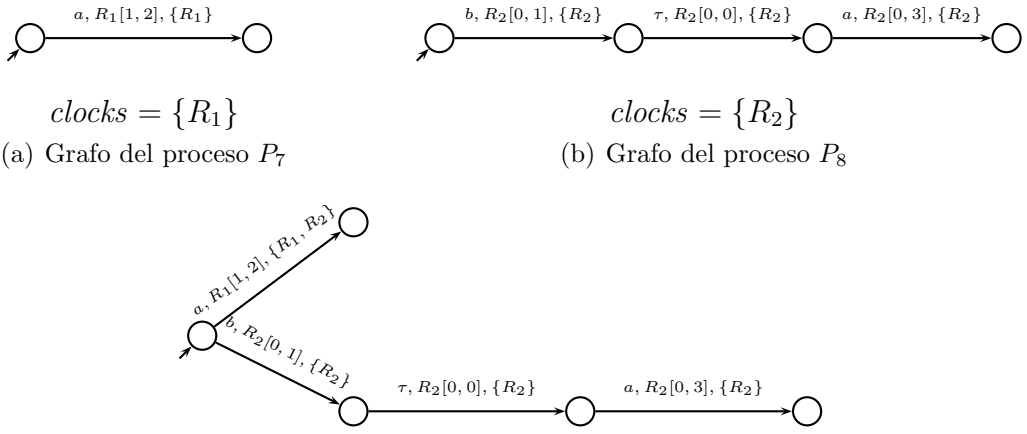
donde $\bar{\lambda}_j$ (con $j = 1, 2, 3$) se define así:

$$\bar{\lambda}_j(e) = \begin{cases} \lambda_{ij}(e) & \forall e \in E_i \cap E \\ \lambda_{ij}(e_i) & \text{si } e = (n_0, n_i),\ \text{con } e_i = (n_{0i}, n_i) \end{cases}$$

siendo λ_{ij} la componente j de λ_i.

El número de nodos del nuevo grafo será $|N_1| + |N_2| - 1$, mientras que el número de arcos será $|E_1| + |E_2|$.

Ejemplo 18 Consideremos el proceso $P_7 = a < 1, 2 >; stop$, cuyo grafo se muestra en la Figura 4.7(a), y el proceso $P_8 = b < 0, 1 >; \tau; a < 0, 3 >; stop$ cuyo grafo se muestra en la Figura 4.7(b). Puesto que ambos procesos cumplen la restricción de que sus nodos iniciales no poseen arcos de entrada, no será necesario aplicar el operador *Unfold*, y el grafo para el proceso $P_9 = P_7 \,\square\, P_8$ será el mostrado en la Figura 4.7(c).



(a) Grafo del proceso P_7

(b) Grafo del proceso P_8

(c) Grafo del proceso P_9

Figura 4.7: Grafo para el proceso del ejemplo 18

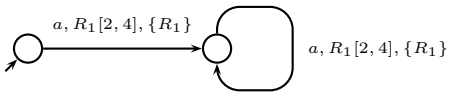
□

Ejemplo 19 Consideremos de nuevo los procesos P_4 y P_5 del ejemplo 17. Como puede observarse en la Figura 4.6(a), el nodo inicial de $G[\![P_4]\!]$ posee arcos entrantes, por lo que debemos aplicar el operador *Unfold* sobre el proceso P_4. Así tendremos que $Unfold(P_4) = a < 1, 2 >; \mu X.a < 1, 2 >; X$ y el grafo para este proceso será el mostrado en la Figura 4.8(a).

El nodo inicial de $G[\![P_5]\!]$ (ver Figura 4.6(b)) no tiene arcos entrantes por lo que no será necesario la aplicación del operador *Unfold*.

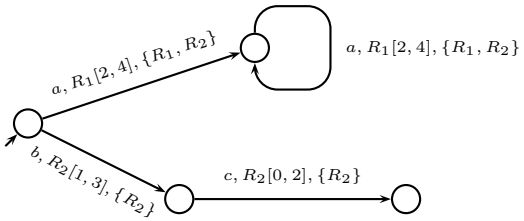
Por tanto, partiendo de estos grafos, el grafo obtenido para el proceso $P_{10} = P_4 \square P_5$ será el mostrado en la Figura 4.8(b)

□



$clocks = \{R_1\}$

(a) Grafo de $Unfold(P_4)$



$clocks = \{R_1, R_2\},\ R_g = R_2$

(b) Grafo de P_{10}

Figura 4.8: Grafo para el proceso del ejemplo 19

4.2.6. Composición paralela

Sea el proceso $P = P_1 \|_{\{A\}} P_2$, donde los procesos P_1 y P_2 tienen asociados respectivamente los grafos:

$$G[\![P_i]\!] = (N_i, E_i, \lambda_i, n_{0i}, clocks_i),\ \text{para } i = 1, 2$$

El grafo para el proceso $P = P_1 \|_A P_2$ se obtiene mediante el producto cartesiano de los grafos P_1 y P_2. El conjunto de relojes del nuevo grafo tendrá los relojes de ambos grafos, además de uno nuevo que será el global. Este reloj será incluido en la etiqueta de todos los arcos del grafo.

Formalmente:

$$G[\![P]\!] = (N, E, \lambda, n_0, clocks)$$

donde:

$$
\begin{array}{ll}
N = & N_1 \times N_2 \\
n_0 = & (n_{01}, n_{02}) \\
E = & E_{11} \cup E_{12} \cup E_{13} \\
\text{donde:} & \\
E_{11} = & \{((n_1, n_2), (n_1', n_2)) \,|\, n_1 \xrightarrow{e_1}_1 n_1', \lambda_{11}(e_1) \notin A,\ n_2 \in N_2\} \\
E_{12} = & \{((n_1, n_2), (n_1, n_2')) \,|\, n_2 \xrightarrow{e_2}_2 n_2', \lambda_{21}(e_2) \notin A,\ n_1 \in N_1\} \\
E_{13} = & \{((n_1, n_2), (n_1', n_2')) \,|\, n_1 \xrightarrow{e_1}_1 n_1',\ n_2 \xrightarrow{e_2}_2 n_2', \lambda_{11}(e_1) = \lambda_{21}(e_2) \in A\} \\
clocks = & clocks_1 \cup clocks_2 \cup \{R_g\},\ \text{t.q. } R_g \notin clocks_1 \cup clocks_2
\end{array}
$$

La función de etiquetado de los arcos será:

$$
\lambda(e) = \begin{cases}
(\lambda_{11}(e_1), \lambda_{12}(e_1), \lambda_{13}(e_1) \cup \{R_g\}) & \text{para } e_1 \in E_1,\ e \in E_{11} \\
(\lambda_{21}(e_2), \lambda_{22}(e_2), \lambda_{23}(e_2) \cup \{R_g\}) & \text{para } e_2 \in E_2,\ e \in E_{12} \\
(a, \lambda_{12}(e_1) \cup \lambda_{22}(e_2), \lambda_{13}(e_1) \cup \lambda_{23}(e_2) \cup \{R_g\}) & \text{para } e \in E_{13}, a = \lambda_{11}(e_1)
\end{cases}
$$

El número de nodos del nuevo grafo será $|N_1| \times |N_2|$ y el número de arcos será del orden de $|E_1| \times |E_2|$, aunque puede ser menor en función del número de sincronizaciones.

Ejemplo 20 Consideremos los procesos $P_{11} = c < 1, 2 >; a < 2, 3 >; stop$, cuyo grafo se muestra en la Figura 4.9(a), y $P_{12} = b < 1, 3 >; \tau; a < 0, 3 >; stop$, cuyo grafo se muestra en la Figura 4.9(b). El grafo para el proceso $P_{13} = P_{11} \|_{\{a\}} P_{12}$ será el mostrado en la Figura 4.9.

□

4.2.7. Ocultamiento

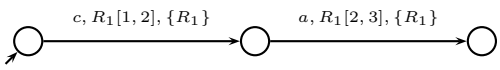
Sea el proceso $P = P_1 \backslash a$, donde $G[\![P_1]\!] = (N_1, E_1, \lambda_1, n_{01}, clocks_1)$ es el grafo asociado a P_1.

El grafo asociado al proceso P podemos definirlo como:

$$G[\![P]\!] = (N_1 \backslash a, E_1, \lambda, n_{01}, clocks_1)$$

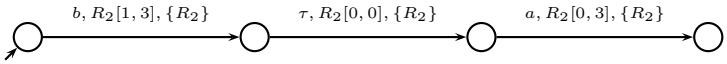
donde la función de etiquetado para los arcos será:

$$
\lambda(e) = \begin{cases}
(\tau, \lambda_{12}(e), \lambda_{13}(e)) & \forall e \in E_1,\ \ \lambda_{11}(e) = a \\
\lambda_1(e) & \text{en caso contrario}
\end{cases}
$$



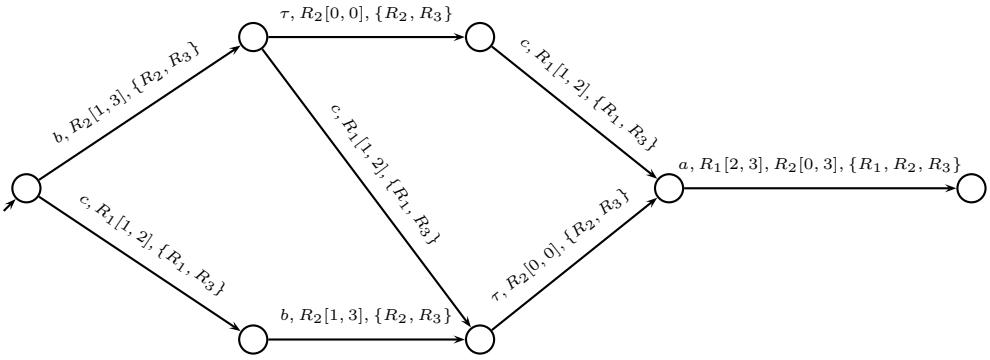
$clocks = \{R_1\}$

(a) Grafo del proceso P_{11}



$clocks = \{R_2\}$

(b) Grafo del proceso P_{12}



$clocks = \{R_1, R_2, R_3\},\ R_g = R_3$

(c) Grafo del proceso P_{13}

Figura 4.9: Grafos correspondientes al ejemplo 20

El término $N_1 \backslash a$ representa los términos de N_1, a los que se les ha añadido en su sintaxis $\backslash a$.

Como puede observarse, para la construcción del grafo asociado al proceso $P \backslash a$, lo que se hace es cambiar las etiquetas de las transiciones etiquetadas con la acción a, de modo que en el nuevo grafo estas transiciones se etiquetan con la acción τ, lo que las convierte en urgentes, de forma similar a lo que ocurría con la semántica operacional.

El tamaño del grafo resultado será el mismo que el del grafo de partida, ya que solamente se cambian algunas etiquetas.

4.2.8. Identificador X

En principio, dado que trabajamos con términos cerrados, no necesitamos introducir un grafo para un identificador. Ahora bien, dado que los identificadores se utilizan para la definición de la recursión, para definir correctamente el grafo del operador recursión necesitamos previamente asociar un grafo a un identificador X.

Así pues, el grafo para el proceso X lo construiremos de forma similar al del proceso *stop*, variando simplemente la etiqueta del nodo.

$$G[\![X]\!] = (\{X\}, \emptyset, \emptyset, X, \{R_g\})$$

4.2.9. Recursión

Sea el proceso $P = \mu X.P_1$ y sea $G[\![P_1]\!] = (N_1, E_1, \lambda_1, n_{01}, clocks_1)$ el grafo construido para el proceso P_1. El grafo asociado al proceso $P = \mu X.P_1$ será:

$$G[\![P]\!] = (N, E, \lambda, n_{01}, clocks_1)$$

donde el conjunto de nodos será el conjunto N_1 menos los nodos etiquetados con X, y donde cada ocurrencia de X en los términos que etiquetan los nodos es reemplazada por P.

$$N = N_1 \setminus \{n \in N_1 \mid term(n) = X\}$$

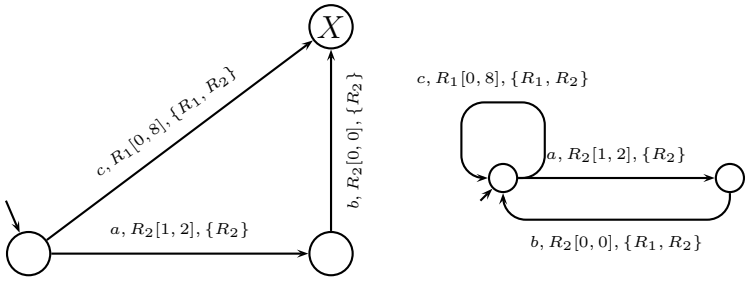
El conjunto de arcos se mantiene, modificando únicamente el nodo destino de aquellos arcos cuyo nodo destino era uno de los eliminados, pasando a ser dicho destino el nodo inicial de $G[\![P_1]\!]$.

$$E = (E_1 \cup \{(n, n_{01}) \mid (n, m) \in E_1 \mid term(m) = X\}) \setminus \{(n, m) \in E_1 \mid term(m) = X\}$$

La función de etiquetado de los arcos se verá modificada, de modo que, para estos arcos cuyo nodo destino se ha modificado, debe cambiarse el conjunto de relojes que sincronizan con el global. El nuevo conjunto para estos arcos será $clocks_1$. Así la función de etiquetado quedará:

$$\lambda(e) = \begin{cases} \lambda_1(e) & \text{si } e \in E_1 \\ (\lambda_{11}(e), \lambda_{21}(e), clocks_1) & \text{si } e = (n, n_{01}) \ \wedge (n, m) \in E_1, term(m) = X \end{cases}$$

Ejemplo 21 Sea el proceso regular $P_{14} = \mu X.(a < 1,2 >; b < 0,0 >; X \,\square\, c < 0,8 >; X)$. El grafo del proceso $P_{14}(X)$ es el mostrado en la Figura 4.10(a). A partir de él obtenemos el grafo para P_{14}, el cual se muestra en la Figura 4.10(b).



$clocks = \{R_1, R_2\},\ R_g = R_2$ $clocks = \{R_1, R_2\},\ R_g = R_2$

(a) Grafo para $P_{14}(X)$ (b) Grafo de P_{14}

Figura 4.10: Grafos correspondientes al ejemplo 21

□

Proposición 1 $G[\![P]\!]$ está bien definido para todo $P \in RTPAL$.

Demostración:

- Términos finitos
  Es trivial.
- Términos Infinitos. En este caso procederemos por inducción estructural.
  - *stop*
    Inmediata.
  - Prefijo temporizado: $P = a < t_1, t_2 >; P_1$
    Mediante la aplicación de la hipótesis de inducción obtenemos el grafo $G[\![P_1]\!]$ para P_1. A partir de éste podemos obtener el grafo $G[\![P]\!]$ añadiendo un nuevo nodo y un nuevo arco.
  - Prefijo de acción urgente: $P = \tau; P_1$
    Inmediata, siguiendo el mismo razonamiento anterior.
  - Operador *wait*: $P = wait(t); P_1$
    Inmediata, siguiendo un razonamiento similar al aplicado para el prefijo temporizado.
  - Elección Externa: $(P = P_1 \Box P_2)$
    Si P_1 ó P_2 fuera recursivo, o los dos, se aplicará la función *Unfold*, que nos proporcionará una representación sintáctica para el proceso para la cual

podríamos aplicar la hipótesis de inducción sobre ambos componentes, para así construir $G[\![P]\!]$.

- Composición Paralela: $P = P_1 \|_A P_2$
  Aplicando la hipótesis de inducción, podemos decir que los grafos $G[\![P_1]\!]$ y $G[\![P_2]\!]$ han sido construidos. A partir de ellos, y mediante el producto cartesiano, podemos obtener el grafo $G[\![P]\!]$.
- Ocultamiento: $P = P_1 \backslash a$
  Aplicando la hipótesis de inducción tenemos que el grafo $G[\![P_1]\!]$ ha sido construido. A partir de él podemos obtener el grafo $G[\![P]\!]$ que será el mismo, pero con algunas etiquetas modificadas.
- Identificador: X
  Inmediata.
- Recursión: $P = \mu X.P_1$
  Si P es un término regular, tenemos que todas las apariciones de X en P_1 deben estar prefijadas, y además en P_1 no aparecerá ni el operador paralelo ni el operador de ocultamiento, lo que hace que en el grafo $G[\![P_1]\!]$ no existan nodos etiquetados con $X \Box Q$, $X \|_A Q$ ni $X \backslash a$. Por tanto, a partir del grafo $G[\![P_1]\!]$ podemos obtener el grafo $G[\![P]\!]$.

□

Ejemplo 22 Consideremos de nuevo el sistema de cruce de un tren descrito en el ejemplo 3 del capítulo 2 y cuya especificación en TPAL aparece en el ejemplo 8.

Aplicando la construcción obtendremos el grafo mostrado en la figura 4.11, el cual ha sido generado con la herramienta TPAL.

□

4.3. Equivalencia

Para poder establecer la correspondencia entre los términos alcanzables según la semántica operacional y los estados alcanzables en el grafo de estados dinámico asociado a un proceso, necesitamos capturar los estados alcanzables desde un nodo por el paso del tiempo. Sin embargo, debemos tener en cuenta que todas las componentes secuenciales de un proceso tienen su propio reloj, y éstos son actualizados por separado. En consecuencia, la evolución en el tiempo de un nodo debe definirse distinguiendo la evolución en el tiempo de cada una de las componentes secuenciales.

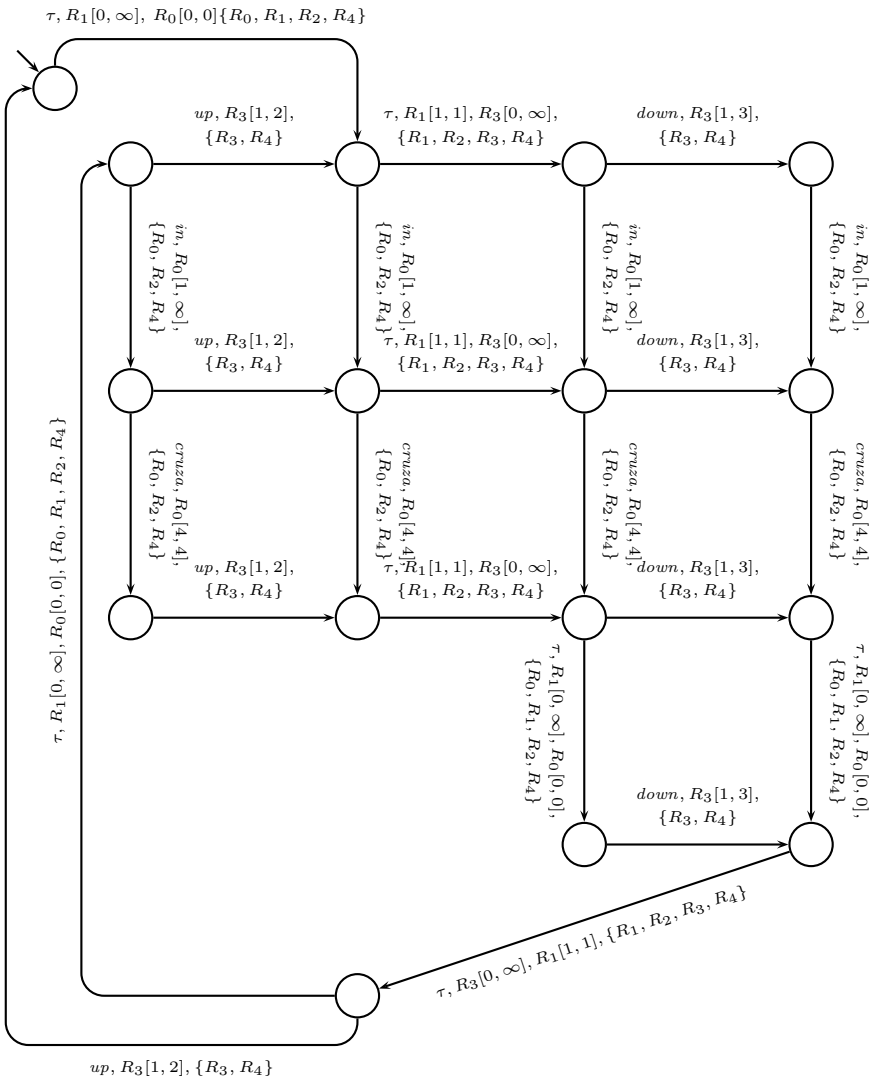


Figura 4.11: Grafo correspondiente al ejemplo 22

Definición 36 (Componentes secuenciales)
Definimos el conjunto de componentes secuenciales con la sintaxis siguiente:

$$S ::= stop \mid a < t_1, t_2 >; P \mid \tau; P \mid wait(t); P \mid S \,\square\, S \mid S\|_A \mid {}_A\|S \mid S\backslash a$$

con $P \in TPAL$.

Entonces, definimos el conjunto de componentes secuenciales de un proceso regular cualquiera P mediante la función:

$$dex : RTPAL \longrightarrow \mathcal{P}_F(S)$$

la cual está definida como sigue:

$$
\begin{aligned}
&dex(stop) = \{stop\}\\
&dex(a < t_1, t_2 >; P) = \{a < t_1, t_2 >; P\}\\
&dex(\tau; P) = \{\tau; P\}\\
&dex(wait(t); P) = \{wait(t); P\}\\
&dex(P_1 \Box P_2) = dex(P_1) \ \Box \ dex(P_2)\\
&dex(P_1 \|_A P_2) = dex(P_1) \|_A \ \cup \ {}_A\| \, dex(P_2)\\
&dex(P \backslash a) = dex(P) \backslash a\\
&dex(\mu X.P) = dex(P\{\mu X.P/X\})
\end{aligned}
$$

□

Así, definimos la función *time_evol* como sigue:

Definición 37 (time_evol)
Dados dos procesos regulares P y Q. Diremos que $P \in time\_evol(Q)$ si y sólo si existe una biyección $\varphi : dex(Q) \longrightarrow dex(P)$ tal que:

$$\forall\, S_Q \in dex(Q)\ \exists\, r \in \mathbb{R}_0^+,\ old(S_Q, r) = \varphi(S_Q)$$

donde la función *old* está definida como sigue:

$$
\begin{aligned}
&old(stop, r) = stop\\
&old(\tau; P, r) = \begin{cases} stop & \text{si } r > 0 \\ \tau; P & \text{si } r = 0 \end{cases}\\
&old(a < t_1, t_2 >; P, r) = \begin{cases} stop & \text{si } r > t_2 \\ a < t_1 \ \dot{-} \ r, t_2 - r >; P & \text{si } r \leq t_2 \end{cases}\\
&old(wait(t); P, r) = \begin{cases} stop & \text{si } r > t \\ wait(t - r); P & \text{si } r \leq t \end{cases}\\
&old(S_1 \Box S_2, r) = old(S_1, r) \,\Box\, old(S_2, r)\\
&old(S\|_A, r) = old(S, r)\|_A\\
&old({}_A\|S, r) = {}_A\|old(S, r)\\
&old(S\backslash a, r) = old(S, r)\backslash a
\end{aligned}
$$

□

Ahora podemos proporcionar dos resultados que establecen la equivalencia entre la semántica operacional y la del grafo.

Teorema 1
Dados dos procesos regulares cualesquiera $P, P' \in RTPAL$, si tenemos que $P \stackrel{<s>}{\Longrightarrow}_t P'$, entonces existe un término $\bar{P}' \in TPAL$ y un nodo $n_{\bar{P}'} \in G[\![P]\!]$ cuyo término correspondiente es $\bar{P}'$ tal que:

$$P' \in time\_evol(\bar{P}')$$

y existe un estado $(n_{\bar{P}'}, C, t)$ en $G[\![P]\!]$ tal que:

$$(n_0, C_0, 0) \stackrel{<s>}{\Longrightarrow}_t (n_{\bar{p}'}, C, t)$$

siendo s la secuencia de acciones y tiempos ejecutadas para alcanzar ese estado.

Demostración:
La demostración de este teorema la realizaremos por inducción estructural sobre los operadores del lenguaje, y para cada uno de los operadores procederemos por inducción sobre la longitud de la secuencia s utilizada para alcanzar el proceso P'.

- *stop*
  Dado el proceso $P = stop$, la única regla de la semántica operacional que puede ser aplicada sobre el proceso P es $P \stackrel{}{\underset{t}{\longmapsto}} P'$, siendo $P' = stop$.

  En el grafo asociado a este proceso solamente tendremos el nodo raíz, en el cual sólo podrá pasar el tiempo según la semántica de los grafos, ya que para ese nodo tendremos que $Min_\tau(n_0) = \infty$.

- Prefijo Temporizado: $P = a < t_1, t_2 >; P_1$

  - Caso base ($|s| = 1$)
    En este caso solamente aplicaremos una regla de transición. Dependiendo de cual sea ésta, podremos distinguir los casos siguientes:

    - Caso b.1: $P \stackrel{a}{\longrightarrow} P_1$
      En el grafo existirá un arco cuya etiqueta es $\lambda(e) = (a, R_g[t_1, t_2], clocks_{P_1})$, que va del nodo raíz del grafo al raíz de $G[\![P_1]\!]$. La única posibilidad de aplicar esta regla es porque $t_1 = 0$, y es el único caso en que podríamos aplicar la regla.
      Entonces, según la semántica del grafo, podremos llegar al nodo n_{P_1} (raíz de P_1) por el arco etiquetado con a en tiempo 0, por lo que el reloj global de P_1 será inicializado a cero. Además se cumplirá que $P_1 \in time\_evol(P_1)$.

- Caso b.2: $P \underset{t}{\longmapsto} P'$
  El proceso P' obtenido tras esta transición será:

  $$P' = a < t_1 - t, t_2 - t >; P_1$$

  el cual cumple que $P' \in time\_evol(P)$.
  En el grafo nos mantendremos en el nodo raíz, en el cual está permitido el paso del tiempo ya que se cumple que $Min_\tau(n_0) = \infty$.
- Caso b.3: $P \underset{t}{\longmapsto} stop$
  La aplicación de esta regla sólo será posible si $P = a < 0, 0 >; P_1$. En el grafo nos encontraremos en el nodo raíz, en el que se permite pasar el tiempo, ya que se cumple $Min_\tau(n_0) = \infty$. También se cumple que $stop \in time\_evol(P)$.

- Caso general ($|s| > 1$)
  Consideraremos la secuencia s como $s = x \cdot s_1$, donde x será una acción o el paso del tiempo. Al igual que antes debemos distinguir la evolución según los casos anteriores.
  - Caso g.1: $P \xrightarrow{a} P_1 \overset{<s_1>}{\Longrightarrow}_t P'$
    Aplicando la hipótesis de inducción sobre el proceso P_1 tendremos que existe un nodo $n_{\bar{P'}}$ en $G[\![P_1]\!]$ tal que $P' \in time\_evol(\bar{P'})$ y existe un estado $(n_{\bar{P'}}, C, t)$ alcanzable en $G[\![P_1]\!]$ mediante la ejecución de s_1 en tiempo t. En el grafo $G[\![P]\!]$, tras realizar la transición correspondiente a la acción a, estaremos en el nodo inicial de $G[\![P_1]\!]$ donde todos los relojes tendrán valor 0. Así, podremos ejecutar la misma secuencia de acciones que en $G[\![P_1]\!]$ para alcanzar el estado $(n_{\bar{P'}}, C, t)$.
  - Caso g.2: $P \underset{t'_1}{\longmapsto} P_1 \overset{<s_1>}{\Longrightarrow}_{t'_2} P'$, y $t'_1 + t'_2 = t$
    Podemos suponer, sin pérdida de generalidad, que la primera regla de s_1 no es de paso de tiempo, y que todo el tiempo transcurrido antes de aplicar esa regla está contabilizado en t'_1. Por tanto supondremos $s_1 = a \cdot s_2$, de modo que $P_1 \xrightarrow{a} P_2 \overset{<s_2>}{\Longrightarrow}_{t'_2} P'$. En primer lugar debemos comprobar que la acción a será ejecutada en un instante $t \in [t_1, t_2]$. Esto es inmediato usando la semántica del grafo. Una vez ejecutada esta acción nos encontraremos en el nodo inicial del grafo $G[\![P_2]\!]$ pero con todos los relojes inicializados a t'_1. Aplicando la hipótesis de inducción y el lema 1 tendremos que $\exists n_{\bar{P'}}$ tal que $(n_{\bar{P'}}, C, t)$ es alcanzable en $G[\![P]\!]$, donde $t = t'_1 + t'_2$.

- Prefijo de acción urgente: $P = \tau; P_1$
  Al igual que hemos realizado para el prefijo temporizado, en este caso procederemos por inducción sobre la longitud de s.

  - Caso base ($|s| = 1$)
    La única regla aplicable es $P \xrightarrow{\tau} P_1$, la cual debe ser aplicada en el instante 0.

    En el nodo inicial del grafo $G[\![P]\!]$ tendremos que $Min_\tau(n_0) = 0$, por lo que no se permitirá el paso del tiempo y deberá ejecutarse esa transición en el instante 0, de modo que el conjunto de relojes $clocks_1$ de P_1 se actualizan a 0.

  - Caso general ($|s| > 1$).
    La secuencia ejecutada será $s = \tau \cdot s_1$. Como se comentó en el caso anterior, la ejecución de τ se realiza en el instante 0 y el conjunto de relojes $clocks_1$ se actualizará a 0. Con la ejecución de esta acción pasaremos al estado inicial del proceso P_1. Aplicando sobre este proceso la hipótesis de inducción tendremos que existe un estado $(n_{\bar{P}'}, C, t)$ alcanzable por el proceso P tal que $P' \in time\_evol(\bar{P}')$.

- Operador wait: $P = wait(t); P_1$
  Si $t = 0$ tendremos que la única regla aplicable a este proceso será $P \xrightarrow{\tau} P_1$ por lo que nos encontraremos en el caso descrito para el operador τ.

  Si $t > 0$ y la longitud de s es 1 tendremos que la única regla aplicable es $P \xmapsto[\Delta t]{} P'$ siendo $P' = wait(t - \Delta t); P_1$ y $\Delta t \leq t$.

  En el grafo asociado a P tendremos que $Min_\tau(n_0) = t$ por lo que se permitirá el paso de esa cantidad de tiempo. Además nos mantendremos en el nodo asociado a P y se cumplirá $P' \in time\_evol(P)$.

  La generalización a secuencias s tales que $|s| > 1$ la realizaremos de forma similar a como lo hicimos en el operador anterior. Tras el paso de t unidades de tiempo tendremos que el proceso P ha evolucionado y ahora será $wait(0); P_1$. A partir de este proceso podemos aplicar el razonamiento que hicimos en el operador anterior con lo que tendremos que existe un estado $(n_{\bar{P}'}, C, t)$ alcanzable por el proceso P tal que $P' \in time\_evol(\bar{P}')$.

- Elección Externa: $P = P_1 \Box P_2$
  La aplicación de la función *Unfold* no afecta al comportamiento del proceso,

ya que esta función solamente extiende la recursión más externa y no afecta a la evolución del proceso según la semántica operacional.

La aplicación de esta función tampoco afectará a la aplicación de la hipótesis de inducción, ya que es una función puramente sintáctica. Por tanto, podemos suponer sin pérdida de generalidad que $P = P_1 \,\square\, P_2$, donde la operación *Unfold* ya ha sido realizada sobre ambos procesos.

- Caso base ($|s| = 1$)
  - Caso b.1: $P \xrightarrow{a} P' = P_{1k}$
    La ejecución de esta transición se produce por la aplicación de la regla $P_1 \xrightarrow{a} P_{1k}$.
    Si aplicamos la hipótesis de inducción sobre el proceso P_1 tendremos que $\exists n_{\bar{P_{1k}}}$ en el grafo $G[\![P_1]\!]$ alcanzable en tiempo $t = 0$ de modo que $P_{1k} \in time\_evol(\bar{P_{1k}})$.
    Este resultado es transferido inmediatamente sobre el proceso P.
  - Caso b.2: $P \xrightarrow{a} P' = P_{2k}$
    Es el caso dual al anterior en el cual la evolución se produce por P_2.
  - Caso b.3: $P \underset{t}{\longmapsto} P' = P_1' \square P_2'$
    La aplicación de esta regla es posible porque $P_1 \underset{t}{\longmapsto} P_1'$ y $P_2 \underset{t}{\longmapsto} P_2'$.
    Aplicando la hipótesis de inducción sobre ambos procesos tendremos que en ambos grafos existe un nodo $n_{\bar{P_i'}}$ que en este caso es el inicial, el cual permite el paso de t unidades de tiempo, de modo que se cumple $P_i' \in time\_evol(\bar{P_i'})$. Si ambos grafos permiten el paso de t unidades de tiempo, es porque se cumple que $Min_\tau(C_{0i}, n_{0i}) \geq t$ y por tanto para el grafo global se cumple que $Min_\tau(C_0, n_0) \geq t$. Así pues podemos concluir que el grafo permite el paso de t unidades de tiempo.
- Caso general ($|s| > 1$)
  - Caso g.1: $P \xrightarrow{a} P_{1k} \overset{<s_1>}{\Longrightarrow}_{t_2} P' = P_1'$
    Supongamos que evolucionamos por P_1, por lo que la evolución será $P_1 \xrightarrow{a} P_{1k} \overset{<s_1>}{\Longrightarrow}_{t_2} P' = P_1'$. Aplicando la hipótesis de inducción sobre este proceso tendremos que $\exists n_{\bar{P_1'}}$ en el grafo $G[\![P_1]\!]$ y éste estará también en el grafo $G[\![P]\!]$. La semántica del grafo G permite la ejecución de la acción a, manteniendo los relojes su valor inicial a 0. La propiedad se concluye aplicando la hipótesis de inducción.

- Caso g.2: $P \xrightarrow{a} P_{1k} \overset{<s_1>}{\Longrightarrow}_{t_2} P' = P'_2$
  Este es el caso dual del anterior.
- Caso g.3: $P \underset{t_1}{\longmapsto} P_{1k} \Box P_{2k} \overset{<s_1>}{\Longrightarrow}_{t_2} P' = P'$
  Consideramos que t_1 totaliza todo el tiempo de espera inicial, siendo la primera acción de s_1 la que resuelve la elección. Esta consideración no resta generalidad a la demostración. Consideremos además que una vez pasado el tiempo la computación ejecutada es de P_1, es decir:

  $$P_{1k} \overset{<s_1>}{\Longrightarrow}_{t_2} P'$$

  Aplicando la hipótesis de inducción tendremos que en el grafo asociado al proceso P_{1k} podemos alcanzar un nodo $n_{\bar{P'}}$ en tiempo t_2 el cual cumple que $P' \in \mathit{time\_evol}(\bar{P'})$. En el grafo $G[\![P]\!]$ el paso de las t_1 unidades de tiempo está permitido y la traza s_1 puede ser ejecutada por lo que ese nodo también es alcanzable en $G[\![P]\!]$.

- Composición paralela: $P = P_1 \|_A P_2$

  - Caso base ($|s| = 1$)
    Distinguiremos tres casos dependiendo de la transición ejecutada.
    - Caso b.1: $P \xrightarrow{a} P',\ a \notin A$
      Consideremos que la evolución se realiza mediante una acción perteneciente al proceso P_1. El mismo razonamiento se haría si la acción fuese de P_2.
      Tendremos que $P_1 \xrightarrow{a} P'_1$ por lo que $P \xrightarrow{a} P'_1 \|_A P_2$. En el grafo $G[\![P_1]\!]$ tendremos un arco etiquetado con $(a, \lambda_{12}(e), \lambda_{13}(e))$ desde el nodo raíz P_1 hasta el nodo $n_{P'_1}$. Aplicando la hipótesis de inducción sobre el proceso P_1 tendremos que esa transición puede ser ejecutada en el grafo, de modo que se alcance el nodo $n_{P'_1}$ y cumpliéndose que $P'_1 \in \mathit{time\_evol}(P'_1)$.
      En el Grafo $G[\![P]\!]$ tendremos una arco etiquetado como el anterior desde el nodo raíz (n_{01}, n_{02}) hasta el nodo (n_1, n_{02}) correspondiente al proceso $P'_1 \|_A P_2$, de modo que $P'_1 \|_A P_2 \in \mathit{time\_evol}(P'_1 \|_A P_2)$.
      No existirán problemas con los tiempos, ya que la acción es ejecutada en tiempo 0.
    - Caso b.2: $P \xrightarrow{a} P',\ a \in A$
      Dados P_1 y P_2 los procesos involucrados en la composición paralela,

para cada uno de ellos tenemos que las siguientes evoluciones son posibles $P_1 \xrightarrow{a} Q_1$ y $P_2 \xrightarrow{a} Q_2$, respectivamente. Por lo tanto podemos aplicar la regla $P \xrightarrow{a} Q_1 \|_A Q_2$. Si aplicamos la hipótesis de inducción a ambos procesos tendremos que en sus respectivos grafos existe un arco desde n_{P_i} hasta n_{Q_i} etiquetado con a, que puede ser ejecutado, de modo que se alcance el nodo asociado al proceso Q_i en tiempo 0, por lo que $Q_i \in time\_evol(Q_i)$. Si trasladamos ésto al grafo de $P = P_1 \|_A P_2$ tendremos que existe un arco desde el nodo (n_{P_1}, n_{P_2}) hasta el nodo (n_{Q_1}, n_{Q_2}) que puede ser ejecutado, y además $Q_1 \|_A Q_2 \in time\_evol(Q_1 \|_A Q_2)$.

- Caso b.3: $P \underset{t}{\longmapsto} P'$ con $P' = Q_1 \|_A Q_2$
  En este caso tendremos que $P_1 \underset{t}{\longmapsto} Q_1$ y $P_2 \underset{t}{\longmapsto} Q_2$.
  Aplicando la hipótesis de inducción a ambos procesos tendremos que G_1 y G_2 permiten el paso del tiempo ya que en ambos grafos se cumple que $t \leq Min_\tau(n_{0i})$.
  Así tenemos que en $G[\![P]\!]$ se cumplirá que $t \leq Min_\tau(n_{01}, n_{02})$, por lo que podemos concluir que en $G[\![P]\!]$ también puede pasar ese tiempo.

- Caso general ($|s| > 1$)

  Para realizar la generalización para secuencias s tales que $|s| > 1$ partiremos la secuencia s en dos subsecuencias, la subsecuencia s_1 que contiene las acciones y retrasos de P_1 y la subsecuencia s_2 que contiene las acciones y retrasos de P_2. La intersección de estas dos subsecuencias contendrá las acciones de sincronización.

  Consideremos la secuencia $s = t_1\ a_1\ t_2\ a_2\ \ldots t_n\ a_n$. donde cada t_i acumula todo el tiempo transcurrido antes de la ejecución de la acción a_i, pudiendo ser alguno de los t_i cero. Así tendremos la secuencia de transiciones siguiente: $P \underset{t_1}{\longmapsto} \bar{P}_1 \xrightarrow{a_1} P'_2 \underset{t_2}{\longmapsto} \bar{P}_2 \xrightarrow{a_2} P'_3 \ \ldots\ \bar{P}_n \xrightarrow{a_n} P'_{n+1} = P'$.

  Dividiendo esta secuencia como hemos comentado anteriormente tendremos que $P_1 \overset{<s_1>}{\Longrightarrow}_t P'_1$ y $P_2 \overset{<s_2>}{\Longrightarrow}_t P'_2$ de modo que $P' = P'_1 \|_A P'_2$, y en cada etapa intermedia $\bar{P}_i = \bar{P}_{i1} \|_A \bar{P}_{i2}$ y $P'_i = P'_{i1} \|_A P'_{i2}$.

  Aplicando la hipótesis de inducción para cada s_i tendremos que ésta es realizable en el grafo $G[\![P_i]\!]$. A partir de esta consideración podemos concluir que la secuencia total s es posible en $G[\![P]\!]$ ya que el conjunto de relojes de $G[\![P_1]\!]$ es distinto al de $G[\![P_2]\!]$, y en su evolución por separado no interfieren unos en otros, de modo que los relojes que no han sido sincronizados mantienen su "hora local". La única consideración viene cuando se produce un paso del tiempo. En este caso, ambos procesos permiten el paso del tiempo, y por lo tanto el grafo resultado lo permite también.

  El reloj global ha sido introducido en todos los nodos por lo que su actualización es correcta.

  Así pues, tenemos que tras la ejecución de la secuencia s estaremos en el nodo $(n_{\bar{P}'_1}, n_{\bar{P}'_2})$ y por la aplicación de la hipótesis de inducción que hicimos sobre las subsecuencias tenemos que $P'_1 \in \mathit{time\_evol}(\bar{P}'_1)$ y $P'_2 \in \mathit{time\_evol}(\bar{P}'_2)$, por lo que usando la definición de *time_evol* podemos concluir que $P' = P'_1 \|_A P'_2 \in \mathit{time\_evol}(\bar{P}'_1 \|_A \bar{P}'_2)$.

- Ocultamiento: $P = P_1 \backslash a$

  Consideremos el grafo $G[\![P_1]\!]$. A partir de éste obtenemos el grafo $G[\![P]\!]$ cambiando las etiquetas de todos los arcos etiquetados con a por τ.

  Aplicando la hipótesis de inducción y la regla R7c es fácil concluir que este cambio no afecta a otras acciones que eran ejecutables en P, es decir, no se deshabilitan acciones en $G[\![P]\!]$ por el carácter urgente de las nuevas τ que aparecen.

Entonces, si tenemos la computación $P_1 \backslash a \stackrel{<s>}{\Longrightarrow}_t P' \backslash a$ tendremos la computación $P_1 \stackrel{<s_1>}{\Longrightarrow}_t P'$ equivalente en P_1; y usando la hipótesis de inducción obtenemos que existe un estado $(n_{\bar{P}'_1}, C, t)$ alcanzable en $G[\![P_1]\!]$, de modo que $P'_1 \in \mathit{time\_evol}(\bar{P}'_1)$. Entonces la misma secuencia s_1 puede ser ejecutada en $[\![P]\!]$ donde las apariciones de a han sido sustituidas por τ.

- Recursión: $P = \mu\, X.Q$

  - Caso base ($|s| = 1$)

    - Caso b.1: $P \stackrel{a}{\longrightarrow} P'$

      Si el proceso P puede realizar la acción a es porque el proceso $Q\{\mu X.Q \;/\; X\}$ la puede ejecutar también. Entonces podemos aplicar la hipótesis de inducción sobre el proceso Q; con lo que obtendremos que existe un nodo $n_{\bar{P}'}$, alcanzable al ejecutar la acción a en tiempo 0 por lo que $P' \in \mathit{time\_evol}(\bar{P}')$.

      Si el nodo alcanzado en el grafo de Q estaba etiquetado con X, en el grafo de P este nodo habrá desaparecido de modo que todos los arcos que a él llegaban ahora llegan al nodo inicial de Q. Por lo tanto, el nodo que se alcanzará en el grafo de P será el nodo inicial.

      Si el nodo estaba etiquetado con otro proceso distinto de X, entonces éste aparecerá en el grafo de P, y será alcanzable en tiempo 0, según la semántica del grafo.

    - Caso b.2: $P \stackrel{t}{\longmapsto} P'$

      En este caso P podrá dejar pasar pasar el tiempo si $Q\{\mu X.Q \;/\; X\}$ lo permite. Aplicando la hipótesis de inducción sobre Q tendremos que alcanzaremos un nodo $n_{\bar{P}'}$, que de hecho es el inicial de Q y se cumple por tanto que $P' \in \mathit{time\_evol}(Q)$.

  - Caso general ($|s| > 1$)

    La secuencia ejecutada s se descompone de la forma siguiente $s = s_1 \;\ldots\; s_n$, donde

    $$P \stackrel{<s_1>}{\Longrightarrow}_{t_1} P_1 \stackrel{<s_2>}{\Longrightarrow}_{t_2} P_2 \Longrightarrow \cdots P_{n-1} \stackrel{<s_n>}{\Longrightarrow}_{t_n} P_n$$

    de modo que se cumple que $t = t_1 + t_2 + \cdots + t_n$ y cada $P_i = \mu X.Q$.

    Esta división está tomada de modo que las secuencias s_i comiencen en el nodo inicial de P. Aplicando la hipótesis de inducción podemos asegurar que cada una de esas subsecuencias es reproducible en el grafo de

P, partiendo del nodo inicial de P, con todos los relojes sincronizados (Lema 1).

□

Podemos observar que el inverso no es cierto en general, porque no todo $P' \in time\_evol(\bar{P}')$ puede ser alcanzado usando la semántica operacional. En su lugar tenemos el siguiente resultado.

Teorema 2 Dado un proceso regular cualquiera $P \in RTPAL$, y su grafo asociado $G[\![P]\!]$ si tenemos una computación $(n_0, C_0, 0) \stackrel{<s>}{\Longrightarrow}_t (n_{\bar{P}'}, C, t)$ en $G[\![P]\!]$, entonces existe un proceso regular $P' \in RTPAL$ tal que $P \stackrel{<s>}{\Longrightarrow}_t P'$, donde $P' \in time\_evol(\bar{P}')$.

Demostración: Al igual que para el teorema anterior, procederemos por inducción estructural sobre los operadores del lenguaje, y para cada uno de ellos por inducción sobre la longitud de la secuencia s.

- *stop*
  En el grafo de este proceso tendremos un único nodo que permitirá el paso del tiempo indefinidamente, de acuerdo con la semántica del mismo, ya que $Min_\tau(n_0) = \infty$. La transición correspondiente en el modelo algebraico queda capturada con la regla $stop \underset{t}{\longmapsto} stop$.

- Prefijo Temporizado
  Sea el proceso $P = a < t_1, t_2 >; P'$ cuyo grafo asociado $G[\![P]\!] = (V, E, \lambda, v_0, clocks)$ posee un arco que parte del nodo inicial, el cual está etiquetado por $(a, R_g[t_1, t_2], clocks)$.
  Procederemos por inducción sobre la longitud de s.

  - Caso base ($|s| = 1$)
    - Caso b.1: $(n_0, C, 0) \stackrel{a}{\longrightarrow} (n_{\bar{P}'}, C, t)$
      Para que esta transición pueda tener lugar deberá cumplirse $t_1 = 0$, y la regla utilizada es $a < 0, t >; P' \stackrel{a}{\longrightarrow} P'$, que nos lleva al proceso P', que cumple trivialmente $P' \in time\_evol(P')$.
    - Caso b.2: $(n_0, C, 0) \underset{t}{\longmapsto} (n_{\bar{P}'}, C, t)$
      Al producirse esta transición por el paso del tiempo, tendremos que el nodo activo sigue siendo el nodo inicial ($n_{\bar{P}'} = n_0$). Según la semántica del grafo se podrá dejar pasar el tiempo indefinidamente, ya que se

cumplirá que $Min_\tau(n_0) = \infty$. Según la semántica operacional podremos aplicar la regla R2b, si $t \leq t_2$, o la regla R2c, si $t > t_2$, para llegar en ambos casos a un proceso P'', que cumple que $P'' \in time\_evol(P)$.

- Caso general ($|s| > 1$)
  - Caso g.1: $(n_0, C, 0) \xrightarrow{a} (n_{P_1}, C, 0) \stackrel{<s_1>}{\Longrightarrow}_t (n_{\bar{P}'}, C, t)$
    Tras la realización de la acción a nos encontramos en el nodo n_{P_1} asociado al proceso P_1 en tiempo 0, a partir del cual se podrá realizar la subtraza s_1. Aplicando la hipótesis de inducción tendremos que existe un proceso P' tal que $P_1 \stackrel{<s_1>}{\Longrightarrow}_t P'$ y $P' \in time\_evol(\bar{P}')$. Así pues podemos concluir que $a < t_1, t_2 >; P_1 \stackrel{<a.s_1>}{\Longrightarrow}_t P'$.
  - Caso g.2: $(n_0, C, 0) \underset{t}{\longmapsto} (n_0, C, t) \stackrel{<s_1>}{\Longrightarrow}_{t_1} (n_{\bar{P}'}, C, t + t_1)$
    Suponemos sin pérdida de generalidad que la primera transición de s_1 es la ejecución de una acción (que será la acción "a"). Tras su ejecución se alcanzará un nodo $n_{\bar{P}'}$, que corresponde al proceso P', y todos los relojes estarán sincronizados a t. Por tanto el estado alcanzado es $(n_{P'}, C', t)$ con $C'(R_i) = t,\ \forall\, R_i$.
    Si tomamos P' y el estado inicial $(n_{P'}, C_0, 0)$, con $C_0(R_i) = 0,\ \forall\, R_i$, podremos aplicar el lema 1 para obtener que

    $$(n_{P'}, C_0, 0) \stackrel{<s_1>}{\Longrightarrow}_{t_1} (n_{\bar{P}'}, C'', t_1)$$

    es realizable en $G[\![P']\!]$.
    Por tanto, puede aplicarse la hipótesis de inducción sobre P' y obtener un P'' tal que $P' \stackrel{<s_1>}{\Longrightarrow}_{t_1} P''$ con $P'' \in time\_evol(\bar{P}')$.
    El resto de la demostración es ya inmediata, pues basta comprobar que el paso de t unidades de tiempo es posible en P, conduciéndonos a "$a < t_1 \dot{-} t, t_2 - t >; P'$", y que en ese momento puede realizarse la acción "a", conduciéndonos a P'. En definitiva obtendremos:

    $$P \stackrel{<t.s_1>}{\Longrightarrow}_{t+t_1} P''$$

- Prefijo de acción urgente: $P = \tau; P_1$
  En el grafo tendremos un arco $n_0 \xrightarrow{\tau} n_1$, donde n_1 corresponde a P_1. Aplicando la semántica operacional tendremos que $\tau; P_1 \xrightarrow{\tau} P_1$, y además trivialmente $P_1 \in time\_evol(P_1)$.

  Aplicando la hipótesis de inducción de forma similar a como se hizo para el prefijo temporizado obtendremos la generalización para secuencias s tales que $|s| > 1$.

- Operador Wait: $P = wait(t); P_1$
  En $G[\![P]\!]$ hay un único arco de salida desde el nodo n_0, el cual tiene una etiqueta $(\tau, R_g[t,t], C)$, lo cual implica que esa transición sólo podrá ser ejecutada después de t unidades de tiempo.

  Si observamos los estados alcanzables, en primer lugar obtenemos la posibilidad de pasar hasta t unidades de tiempo, estos estados son de la forma (n_0, C_0, t'), con $0 \leq t' \leq t$.

  Por la regla *R4a* obtendremos $P' = wait(t - t'); P_1$ y $P' \in time\_evol(P)$ de forma inmediata.

  Tras t unidades de tiempo la única forma de evolucionar en el grafo es ejecutando τ, y el estado alcanzado es (n_1, C_1, t), donde n_1 corresponde a P_1.

  Así, la correspondiente computación en el álgebra es $P \stackrel{\tau}{\Longrightarrow}_t P_1$, y obviamente $P_1 \in time\_evol(P_1)$.

  Para el caso general ($|s| > 1$) se razona de forma similar a los casos anteriores, aplicando la hipótesis de inducción.

- Elección Externa: $P = P_1 \Box P_2$
  Suponemos que el operador *Unfold* ya ha sido aplicado sobre los procesos P_1 y P_2.

  - Caso base ($|s| = 1$)
    Según la transición realizada tendremos los dos casos siguientes:

    - Caso b.1: $(n_0, C_0, 0) \stackrel{}{\underset{t}{\longmapsto}} (n_{\bar{P}'}, C_0, t)$
      En este caso $n_{\bar{P}'} = n_0$. Si se produce este paso del tiempo, es porque ambos grafos, $G[\![P_1]\!]$ y $G[\![P_2]\!]$, permiten el paso de t unidades de tiempo en su nodo inicial. Aplicando la hipótesis de inducción en ambos grafos tendremos que $P_i \stackrel{}{\underset{t}{\longmapsto}} P_i'$, $i = 1, 2$, y por tanto: $P_1 \Box P_2 \stackrel{}{\underset{t}{\longmapsto}} P_1' \Box P_2'$, cumpliéndose además en este caso que $P_1' \Box P_2' \in time\_evol(\bar{P'}_1 \Box \bar{P'}_2)$. Nótese que en general no puede concluirse que $P_1' \Box P_2' \in time\_evol(\bar{P'}_1 \Box \bar{P'}_2)$ del hecho de que $P_i' \in time\_evol(\bar{P'}_i)$. Eso sólo puede concluirse si el retraso es el mismo para ambos, como nos ocurre en este caso.

    - Caso b.2: $(n_0, C_0, 0) \stackrel{a}{\longrightarrow} (n_{\bar{P}'}, C_0', 0)$
      En este caso el nodo $n_{\bar{P}'}$ pertenece o bien a $G[\![P_1]\!]$ o bien a $G[\![P_2]\!]$, por lo que esta transición puede ser ejecutada sobre uno de los grafos. Sin

pérdida de generalidad, consideraremos que es ejecutada por $G[\![P_1]\!]$. Si aplicamos la hipótesis de inducción sobre el grafo correspondiente tendremos que $P_1 \xrightarrow{a} P_1'$, de modo que $P_1' \in \mathit{time\_evol}(\bar{P}')$. Finalmente, por la regla R5a: $P_1 \Box P_2 \xrightarrow{a} P_1'$.

- Caso general ($|s| > 1$)
  La secuencia s puede adoptar 2 formas, que nos permiten distinguir los 2 casos siguientes:
  - Caso g.1 ($s = a.s_1$)
    Este caso es similar al caso b.2. con la única diferencia de la longitud de la secuencia. Siguiendo el mismo razonamiento concluiremos que la propiedad se cumple.
  - Caso g.2 ($s = t.s_1$)
    Se razona como en el caso anterior, aplicando la hipótesis de inducción al proceso que finalmente realiza s_1.

- Composición paralela.
  Sea el proceso $P = P_1 \|_A P_2$, cuyo grafo asociado es $G[\![P]\!] = (V, E, \lambda, v_0, \mathit{clocks})$.
  - Caso base ($|s| = 1$)
    - Caso b.1: $(n_0, C, 0) \xrightarrow{a} (n_{\bar{P}'}, C', 0)$ $a \notin A$
      Como $a \notin A$ tendremos que esta transición puede ser realizada por uno de los grafos, $G[\![P_1]\!]$ ó $G[\![P_2]\!]$, y aplicando la hipótesis de inducción obtendremos $P_1 \xrightarrow{a} P_1'$, con $P_1' \in \mathit{time\_evol}(P_1')$, ó $P_2 \xrightarrow{a} P_2'$, con $P_2' \in \mathit{time\_evol}(P_2')$, dependiendo del grafo en el que se ejecutase la transición. Por tanto, mediante la aplicación de las reglas R8a ó R8b tendremos: $P \xrightarrow{a} P_1' \|_A P_2$, con $P_1' \|_A P_2 \in \mathit{time\_evol}(P_1' \|_A P_2)$ ó $P \xrightarrow{a} P_1 \|_A P_2'$, con $P_1 \|_A P_2' \in \mathit{time\_evol}(P_1 \|_A P_2')$.
    - Caso b.2: $(n_0, C, 0) \xrightarrow{a} (n_{\bar{P}'}, C', 0)$ $a \in A$
      Si hemos podido realizar la transición en $G[\![P]\!]$ es porque en el grafo $G[\![P_1]\!]$ tendremos una transición $(n_{01}, C_1, 0) \xrightarrow{a} (n_{P_1'}, C_1', 0)$, y en el grafo $G[\![P_2]\!]$ tendremos una transición $(n_{02}, C_2, 0) \xrightarrow{a} (n_{P_2'}, C_2', 0)$.
      Aplicando la hipótesis de inducción sobre ambos grafos tendremos que $P_1 \xrightarrow{a} P_1'$ y $P_2 \xrightarrow{a} P_2'$. Aplicando la regla R6c tendremos que $P \xrightarrow{a} P_1' \|_A P_2'$. Además, trivialmente se cumple $P_1' \|_A P_2' \in \mathit{time\_evol}(P_1' \|_A P_2')$.
    - Caso b.3: $(n_0, C, 0) \underset{t}{\longmapsto} (n_0, C, t)$
      En el grafo $G[\![P]\!]$ tendremos una transición $(n_0, C, 0) \underset{t}{\longmapsto} (n_{\bar{P}'}, C, t)$.

Ésto se podrá realizar porque en el nodo $n_0 = (n_{01}, n_{02})$ se cumple que $t \leq Min_\tau(n_{01}, n_{02})$, por lo que en el grafo $G[\![P_1]\!]$ se cumplirá que $t \leq Min_\tau(n_{01})$ y en el grafo $G[\![P_2]\!]$ se cumplirá que $t \leq Min_\tau(n_{02})$. Así pues pueden pasar t unidades de tiempo en $G[\![P_1]\!]$ y $G[\![P_2]\!]$.

De, ahí, aplicando la hipótesis de inducción a ambos grafos tendremos $P_1 \underset{t}{\longmapsto} P_1'$ y $P_2 \underset{t}{\longmapsto} P_2'$; y por tanto $P \underset{t}{\longmapsto} P_1' \|_A P_2'$. Además, $P_1' \|_A P_2' \in time\_evol(P_1 \|_A P_2)$.

- Caso general ($|s| > 1$)

  Al igual que hicimos para demostrar el teorema 1 dividiremos la secuencia s en dos subsecuencias, s_1 compuesta por las transiciones ejecutadas en $G[\![P_1]\!]$, y s_2 compuesta por las transiciones ejecutadas por $G[\![P_2]\!]$.

  Aplicando la hipótesis de inducción a ambos tendremos que $P_1 \overset{<s_1>}{\Longrightarrow}_t P_1'$ tal que $P_1' \in time\_evol(\bar{P}_1')$ y $P_2 \overset{<s_2>}{\Longrightarrow}_t P_2'$ tal que $P_2' \in time\_evol(\bar{P}_2')$.

  Si consideramos la composición paralela de P_1 y P_2 podremos intercalar las acciones de s_1 y de s_2 en el orden en el que aparecían en s y por tanto tendremos que $P \overset{<s>}{\Longrightarrow}_t P_1' \|_A P_2'$.

  Probemos ahora que se cumple $P_1' \|_A P_2' \in time\_evol(\bar{P}_1' \|_A \bar{P}_2')$.

  Como $P_i' \in time\_evol(\bar{P}'), i = 1, 2$, entonces:

  $$\exists\, \varphi_i : dex(\bar{P}_i') \longrightarrow dex(P_i')$$

  en las condiciones de la definición 37.

  Para probar que $P_1' \|_A P_2' \in time\_evol(\bar{P}_1' \|_A \bar{P}_2')$ tenemos que definir una biyección:

  $$\varphi_{12} : dex(\bar{P}_1') \|_A dex(\bar{P}_2') \longrightarrow dex(P_1') \|_A dex(P_2')$$

  Por la definición de componente secuencial tenemos:

  $$dex(\bar{P}_1' \,\|_A\, \bar{P}_2') = dex(\bar{P}_1') \|_A \;\cup\; {}_A\|dex(\bar{P}_2')$$
  $$dex(P_1' \,\|_A\, P_2') = dex(P_1') \|_A \;\cup\; {}_A\|dex(P_2')$$

  Entonces podemos definir la nueva biyección como:

  $$\varphi_{12}(S \,\|_A) = \varphi_1(S) \,\|_A$$
  $$\varphi_{12}({}_A\|S) = {}_A\|\varphi_2(S)$$

  y tomando:

- $\forall\, S \parallel_A \in dex(\bar{P}'_1 \parallel_A \bar{P}'_2)$ tenemos que $S \in dex(\bar{P}'_1)$, de modo que $\exists\, r$ tal que $old(S, r) = \varphi_1(S)$. De ahi, según la definición de *old*:

  $$old(S\|_A, r) = old(S, r)\|_A = \varphi_1(S)\|_A$$

  como queríamos demostrar.
- Para ${}_A\|S$ podemos seguir un razonamiento analogo.

- Ocultamiento.
  Sea $P = P_1\backslash a$ en cuyo grafo $G[\![P]\!]$ se ejecuta la secuencia de transiciones s, la cual nos permite alcanzar el estado $(n_{\bar{P}'\backslash a}, C, t)$. Entonces en el grafo $G[\![P_1]\!]$ tendremos la secuencia correspondiente s' la cual contiene las mismas acciones que s y en los mismos tiempos, pero las τ que corresponden a acciones ocultadas ahora son reemplazadas por a.

  Aplicando la hipótesis de inducción obtendremos que existe un proceso P' tal que $P_1 \stackrel{s'}{\Longrightarrow}_t P'$ y $P' \in time\_evol(\bar{P}')$. Aplicando la semántica operacional obtendremos que $P_1\backslash a \stackrel{s}{\Longrightarrow}_t P'\backslash a$, para lo cual debemos comprobar que el ocultamiento de la acción a no afecta a la posible ejecución de la secuencia s, para lo cual procederemos por reducción al absurdo.

  Supongamos que s' tiene la forma siguiente: $s' = s_1 \cdot t \cdot b \cdot s'_2$, con $b \neq a$ y $t > 0$, y que exista otra secuencia $s'' = s_1 \cdot a \cdot s''_2$ que es también posible con arreglo a la semántica operacional. En este caso, al trasladar s'' a $G[\![P]\!]$ ocultando las acciones "a", lo que nos encontraremos es que no puede pasar el tiempo, pues hay una τ permitida inmediatamente, por lo que la secuencia s, obtenida a partir de s' no sería posible, lo cual es una contradicción.

- Recursión: $P = \mu X.Q$
  Consideremos la computación $(n_0, C, 0) \stackrel{<s>}{\Longrightarrow}_t (n_{\bar{P}'}, C', t)$.

  Dividimos la secuencia s de la forma siguiente: $s = s_1\ s_2\ \ \ldots\ s_n$; tal que $(n_0, C_{i-1}, t_{i-1}) \stackrel{s_i}{\Longrightarrow}_{\Delta t_i} (n_0, C_i, t_i)$ con $t_i = t_{i-1} + \Delta t_i$, $C_i(R_j) = t_i$, $\forall\, R_j$, y $n \geq 1$; es decir cada s_i nos lleva desde el nodo inicial de nuevo a ese nodo inicial en un tiempo $\Delta\, t_i$.

  Cada una de estas secuencias s_i serán también secuencias de $G[\![Q\{\mu X.Q/X\}]\!]$, por lo que, aplicando el lema 1 y la hipótesis de inducción para cada una de ellas tendremos que $Q \stackrel{<s_i>}{\Longrightarrow}_{t_i} Q_i$, $i = 1, \ldots n$; con $Q_i \in time\_evol(X)$, $i = 1, \ldots n-1$ y $Q_n \in time\_evol(\bar{P}')$. En este caso se nos plantea en principio un problema técnico con la definición de *time_evol*, ya que esta función no está definida para

X. No obstante, este problema puede resolverse fácilmente, tomando en este caso la misma definición que para *stop*.

Así pues tendremos que $P \stackrel{s_1}{\Longrightarrow}_{t_1} Q_1 \stackrel{s_2}{\Longrightarrow}_{t_2} Q_2 \Longrightarrow \ldots \stackrel{s_n}{\Longrightarrow}_{t_n} Q_n$ por lo que podemos concluir $P \stackrel{s}{\Longrightarrow}_t Q_n$, con $Q_n \in time\_evol(\bar{P}')$.

□

4.4. Reducción de grafos

Uno de los principales inconvenientes que tiene la traducción del álgebra a grafos de estados dinámicos es el problema de la explosión de estados. Esta explosión se produce principalmente por la construcción del operador paralelo, la cual, al basarse en un producto cartesiano, introduce una gran cantidad de nuevos nodos, algunos de los cuales no son alcanzables ya que no están conectados por ningún camino con el nodo inicial del grafo, y otros que aún estando conectados no serán alcanzables debido a las restricciones temporales. En el ejemplo siguiente se muestra este problema.

Ejemplo 23 Consideremos los procesos $P_1 = a < 1,2 >; b < 2,4 >; d < 0,0 > ; stop$ y $P_2 = c < 2,4 >; b < 0,2 >; stop$, cuyos grafos son, respectivamente, los mostrados en las Figuras 4.12(a) y 4.12(b). Al construir el grafo para el proceso $P = P_1 \|_{\{b\}} P_2$, como puede observarse en la Figura 4.12(c), el número de nodos del grafo se incrementa considerablemente, apareciendo diversos nodos que nunca serán alcanzables.

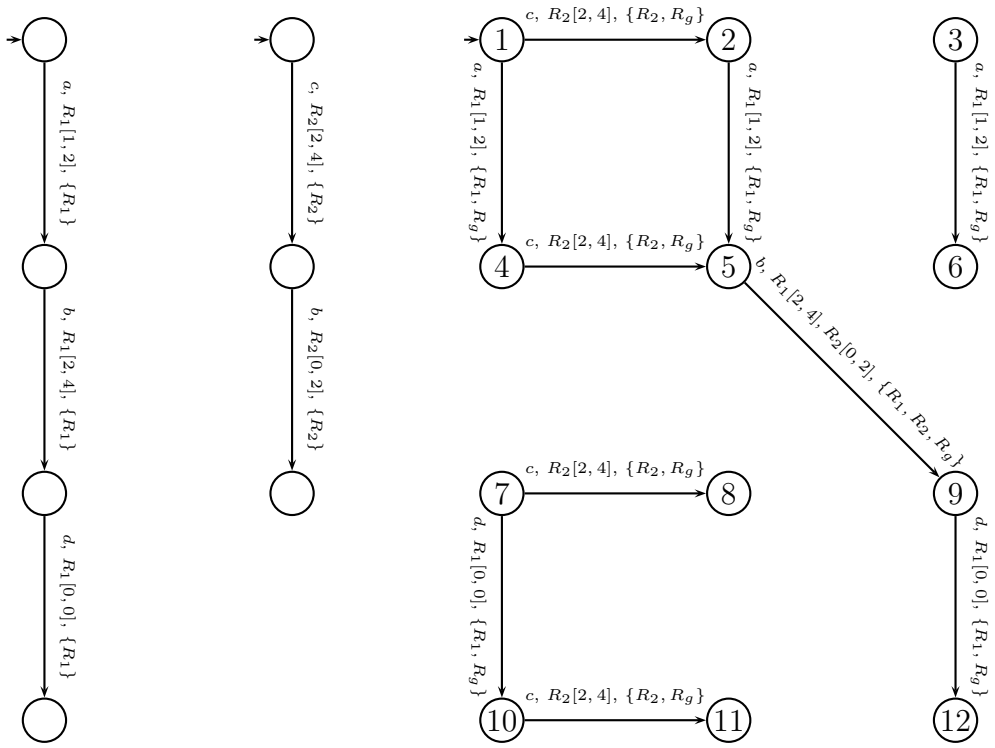
□

Con el fin de intentar paliar este problema, en esta sección presentamos algunos algoritmos, que aplicados a los grafos obtenidos nos permitirán simplificar y reducir los mismos.

Como paso previo, necesitamos introducir una relación de equivalencia entre grafos, con la cual formalizaremos adecuadamente estos algoritmos de reducción.

Definición 38 (Equivalencia $G \sim G'$)
Sean G y G' dos grafos definidos sobre el mismo conjunto de acciones Act.

Diremos que G y G' son equivalentes, y lo representaremos por $G \sim G'$, si y sólo si para toda computación $s_0 \stackrel{<w>}{\Longrightarrow}_t s$ en el grafo G, existe una computación $s'_0 \stackrel{<w>}{\Longrightarrow}_t s'$



(a) Grafo de P_1 (b) Grafo de P_2 (c) Grafo de $P_1 \|_{\{b\}} P_2$

Figura 4.12: Ejemplo de explosión de estados

en el grafo G' y a la inversa, es decir, para toda computación $s'_0 \stackrel{<w>}{\Longrightarrow}_t s'$ en el grafo G' existe una computación $s_0 \stackrel{<w>}{\Longrightarrow}_t s$ en el grafo G.

□

La relación $\sim$ es tan solo una equivalencia de trazas temporizadas, la cual en algunos casos suele ser una equivalencia muy débil ya que no permite diferenciar comportamientos distintos, como por ejemplo, los obtenidos en las álgebras clásicas como consecuencia de ambos operadores de elección (interna y externa). En nuestro caso este problema no se presenta, y para nuestros propósitos es suficiente esta noción de equivalencia en la que consideramos que dos grafos son equivalentes si reconocen el mismo conjunto de palabras temporizadas.

Proposición 2 $G \sim G'$ es una relación de equivalencia.

Demostración: Para demostrar que $\sim$ es una relación de equivalencia deberemos demostrar que cumple las propiedades reflexiva, simétrica y transitiva.

- Reflexiva
  Es obvio que se cumple G $\sim$ G ya que cada computación está relacionada con ella misma.

- Simétrica
  Si se cumple que $G \sim G'$, se cumplirá que para toda computación $s_0 \overset{<w>}{\Longrightarrow}_t s$ en G, existirá una computación $s'_0 \overset{<w>}{\Longrightarrow}_t s'$ en G' y a la inversa, es decir, para toda computación $s'_0 \overset{<w>}{\Longrightarrow}_t s'$ en G' existe una computación $s_0 \overset{<w>}{\Longrightarrow}_t s$ en G. Por tanto $G' \sim G$.

- Transitiva
  Si se cumple que $G \sim G'$ tendremos que para toda computación $s_0 \overset{<w>}{\Longrightarrow}_t s$ en G existe una computación $s'_0 \overset{<w>}{\Longrightarrow}_t s'$ en G'.

  Además, como se cumple que $G' \sim G''$ tendremos que para toda computación $s'_0 \overset{<w>}{\Longrightarrow}_t s'$ en G' existe una computación $s''_0 \overset{<w>}{\Longrightarrow}_t s''$ en G''.

  Así pues podemos concluir que para toda computación $s_0 \overset{<w>}{\Longrightarrow}_t s$ en G existe una computación $s''_0 \overset{<w>}{\Longrightarrow}_t s''$ en G''. De ahí: $G \sim G''$.

  El razonamiento en sentido inverso es análogo.

□

4.4.1. Reducción

A continuación presentaremos algunas reducciones estructurales, que eliminarán del grafo algunos nodos y arcos que no podrán ser ejecutados en ningún momento. También estableceremos algunos casos particulares en los cuales el grafo puede ser simplificado de acuerdo con la relación de equivalencia definida anteriormente.

Eliminación de nodos no alcanzables

Una primera y muy simple reducción aplicable a los grafos es la eliminación de los nodos aislados del nodo inicial, es decir, aquéllos que no están conectados con el nodo inicial del grafo por ningún camino. Junto a la eliminación de estos nodos, se eliminarán todos los arcos cuyo nodo inicial es alguno de los eliminados.

Definición 39 (Nodos no-alcanzables)
Sea $G = (N, E, \lambda, n_0, \mathit{clocks})$ un grafo de estados dinámico cualquiera.

Definimos el conjunto de nodos no alcanzables desde el nodo inicial del grafo G como:

$$Unreach(G) = \{n \in N |\ \nexists \text{ camino } n_0 \cdots n\}$$

□

Definición 40 (Arcos no-alcanzables)
Sea $G = (N,\ E,\ \lambda,\ n_0,\ clocks)$ un grafo de estados dinámico cualquiera.

Definimos el conjunto de arcos no alcanzables de G como el conjunto de arcos cuyo nodo origen es uno no alcanzable.

$$Unreach(E) = \{e \in E \mid n \xrightarrow{e} n',\ n \in Unreach(G)\}$$

□

Una vez definidos estos dos conjuntos podemos definir el grafo reducido de G de la forma siguiente:

$$\text{Reduc(G)} = (N \setminus Unreach(G),\ E \setminus Unreach(E),\ n_0,\ clocks)$$

Proposición 3 Para todo grafo de estados dinámicos G, se cumple:

$$G \sim \text{Reduc(G)}$$

Demostración: Es inmediata.

□

Eliminación de nodos no alcanzables en el tiempo

El método de reducción expuesto anteriormente puede ser generalizado de modo que se eliminen aquellos nodos no alcanzables en el tiempo, que son aquellos nodos que nunca podrán ser alcanzados debido a que las condiciones temporales establecidas en los arcos que permiten alcanzarlos nunca se cumplirán.

Con el objetivo de localizar estos nodos no alcanzables, aplicaremos un algoritmo que nos permita calcular unas cotas temporales para los instantes de tiempo en los que ese nodo puede ser alcanzado. De acuerdo con las cotas obtenidas podremos decidir si ese nodo será alcanzable o no.

Algoritmo de cálculo de cotas temporales

Sea $G = (N, E, \lambda, n_0, \mathit{clocks})$ un grafo de estados dinámico cualquiera y s un camino en el grafo que nos lleva a un nodo n' cualquiera.

Mediante la aplicación de este algoritmo obtendremos una cota inferior, $\mathit{MinR}(s, R_j)$, y una cota superior, $\mathit{MaxR}(s, R_j)$, para el valor de cada uno de los relojes del grafo tras la ejecución de la secuencia s. También se calcula una cota inferior $\mathit{MinD}(s, R_j)$ y una cota superior $\mathit{MaxD}(s, R_j)$ para la diferencia existente entre el valor del reloj R_j y el reloj global R_g tras ejecutar la secuencia s.

Debido a que siempre se cumplirá que $R_j \leq R_g$ tendremos que las cotas para las diferencias entre los relojes cumplirán $\mathit{MinD}(s, R_j) \leq 0$ y $\mathit{MaxD}(s, R_j) \leq 0$.

El algoritmo planteado es un algoritmo iterativo, el cual calculará estos valores para un determinado nodo a partir de los valores obtenidos para el nodo anterior en el camino seguido. Así, el algoritmo será:

1. Inicialización
   La primera secuencia ejecutada es la secuencia vacía: $s_0 = \epsilon$, mediante la cual nos mantenemos en el nodo inicial del grafo, en el cual los valores que tendremos serán:
   $$\mathit{MinR}(s_0, R_j) = 0$$
   $$\mathit{MaxR}(s_0, R_j) = 0$$
   $$\mathit{MinD}(s_0, R_j) = 0$$
   $$\mathit{MaxD}(s_0, R_j) = 0$$

2. Recorrido
   Tras ejecutar una secuencia s se habrá alcanzado el nodo n, en el cual supondremos que para cada uno de los relojes R_j los valores obtenidos son $\mathit{MinR}(s, R_j)$, $\mathit{MaxR}(s, R_j)$, $\mathit{MinD}(s, R_j)$ y $\mathit{MaxD}(s, R_j)$.

   A partir de n podemos alcanzar el nodo n' mediante la ejecución de la transición e, cuya etiqueta es $\lambda(e) = (a, r', S)$. Así, los valores de las cotas para cada uno de los relojes R_j tras la secuencia $s' = s \cdot e$ lo calcularemos de la forma siguiente:

   2.1 Calculamos una cota superior para $\mathit{Min}_\tau(n, C)$:

$$\Delta_\tau = \begin{cases} \infty\ , & \text{si } \nexists e',\ n \xrightarrow{e'},\ \lambda_1(e') = \tau \\ \text{mín}\{\ \underset{k}{\text{máx}}\ \{\Psi_\tau(n, e', R_k)/r'(R_k) \text{ está definida, siendo } r' = \lambda_2(e')\}\} & \\ \underset{\lambda_1(e') = \tau}{e', n \xrightarrow{e'}} & \end{cases}$$

donde

$$\Psi_\tau(n, e', R_k) = \text{máx}\{0\ ,\ \text{mín}\{\alpha_k + MaxD(s, R_k) \\ MaxR(s, R_k) + \alpha_k - MinR(s, R_g)\}\ \}$$

Este valor se calcula para cada uno de los relojes R_k para los que exista una restricción en las transiciones etiquetadas con τ, que parten del nodo n. La etiqueta de estas transiciones será $\lambda(e') = (\tau, r', S')$ y $r'(R_k) = [\alpha_k, \beta_k]$.

2.2 De forma similar calculamos una cota inferior, que denotaremos con γ, para $Min(n, C, e)$ y una cota superior para $Max(n, C, e)$, a la que denotaremos con δ. Así tendremos que:

$$\gamma = \underset{k}{\text{máx}}\ \{\Psi(n, e, R_k)/r'(R_k) \text{ está definida}\}$$
$$\delta = \underset{k}{\text{mín}}\ \{\Phi(n, e, R_k)/r'(R_k) \text{ está definida}\}$$

donde para cada reloj que tiene definida $r'(R_k)$ tendremos que:

- $\Psi(n, e, R_k) = \text{máx}\{0\,,\ \text{máx}\{\alpha_k + MinD(s, R_k) \\ MinR(s, R_k) + \alpha_k - MaxR(s, R_g)\}\}$
- $\phi(n, e, R_k) = \text{mín}\{\beta_k + MaxD(s, R_k), \\ MaxR(s, R_k) + \beta_k - MinR(s, R_g)\}$

2.3 Para finalizar, calcularemos las cotas para el nuevo nodo como:

- $MinR(s', R_j) = \begin{cases} MinR(s, R_j) & \text{si } R_j \notin S \\ MinR(s, R_g) + \gamma & \text{si } R_j \in S \end{cases}$
- $MaxR(s', R_j) = \begin{cases} MaxR(s, R_j) & \text{si } R_j \notin S \\ MaxR(s, R_g) + \text{mín}\{\delta, \Delta_\tau\} & \text{si } R_j \in S \end{cases}$
- $MinD(s', R_j) = \begin{cases} 0 & \text{si } R_j \in S \\ \text{máx}\{MinR(s, R_j) - MaxR(s, R_g), \\ \quad MinD(s, R_j)\} - \text{mín}\{\delta, \Delta_\tau\} & \text{si } R_j \notin S \end{cases}$
- $MaxD(s', R_j) = \begin{cases} 0 & \text{si } R_j \in S \\ \text{mín}\{MaxR(s, R_j) - MinR(s, R_g) - \gamma, \\ \quad MaxD(s, R_j) - \gamma\} & \text{si } R_j \notin S \end{cases}$

Proposición 4 Sea $G = (N, E, \lambda, n_0, \mathit{clocks})$ un grafo de estados dinámico cualquiera, (n, C, t) un estado alcanzable en el mismo y $s = n_1 \cdots n_m$ un camino recorrido en G hasta llegar al estado (n, C, t), donde $n = n_m$.

Entonces $\forall R_j \in \mathit{clocks}$ se cumple:

(I) $\mathit{MinR}(s, R_j) \leq C(R_j) \leq \mathit{MaxR}(s, R_j)$

(II) $\mathit{MinD}(s, R_j) \leq C(R_j) - C(R_g) \leq \mathit{MaxD}(s, R_j)$.

Demostración: Para demostrar esta proposición procederemos por inducción sobre la longitud de la secuencia s ejecutada para alcanzar el nodo n.

- Caso base: $(s = \epsilon)$
  Es trivial.

- Caso general: $(s = s_{m-1} \cdot a)$
  Suponemos cierta la proposición tras haber ejecutado la secuencia s_{m-1}, la cual nos ha llevado al estado (n_{m-1}, C', t'). Comprobaremos que la proposición se cumple tras ejecutar la secuencia s, la cual nos lleva al estado (n, C, t). Para ello seguiremos los pasos siguientes:

  a) En primer lugar probaremos que en el nodo n_{m-1} se cumple que Δ_τ es una cota superior de $\mathit{Min}_\tau(n_{m-1}, C')$, es decir, $\mathit{Min}_\tau(n_{m-1}, C') \leq \Delta_\tau$.
  Para comprobar que esto se cumple, podemos distinguir dos casos:

  - No existen arcos etiquetados con τ que parten del nodo n_{m-1}.
    Tendremos entonces que $\Delta_\tau = \infty$ y $\mathit{Min}_\tau(n_{m-1}, C') = \infty$, por lo que se cumple.
  - Existe algún arco que parte de n_{m-1} etiquetado con τ.
    Aplicando la definición de Min_τ y de $\mathit{Min}(n, C, e)$ para el estado en el que se encuentra el grafo tendremos:

$$\mathit{Min}_\tau(n_{m-1}, C') = \min_{\substack{e \\ n \xrightarrow{e} \\ \lambda_1(e) = \tau}} \{ \max_{\substack{R_j \\ r(R_j) = [\alpha_j, \beta_j]}} \{C'(R_j) + \alpha_j \dot{-} C'(R_g)\}\} \tag{4.1}$$

    Para cada uno de los arcos e tal que $\lambda_1(e) = \tau$, y para cada uno de los relojes R_j que tiene definida una restricción en ese arco, aplicando la definición de $\dot{-}$ tendremos:

$$C'(R_j) + \alpha_j \dot{-} C'(R_g) = \text{máx}\{0, C'(R_j) + \alpha_j - C'(R_g)\}$$

Por la hipótesis de inducción sabemos que se cumplen las siguientes desigualdades:

$$C'(R_j) \leq MaxR(s_{m-1}, R_j) \text{ y } C'(R_g) \geq MinR(s_{m-1}, R_g) \qquad (4.2)$$

por lo que aplicando esta desigualdades en la expresión anterior tendremos:

$$\begin{aligned} C'(R_j) + \alpha_j \dot{-} C'(R_g) \leq \quad & \text{máx}\{0, \\ & MaxR(s_{m-1}, R_j) + \alpha_j - MinR(s_{m-1}, R_g)\} \end{aligned}$$

Si consideramos la discrepancia para R_j, aplicando la hipótesis de inducción y la definición del operador $\dot{-}$, tendremos:

$$C'(R_j) + \alpha_j \dot{-} C(R_g) \leq \text{máx}\{0, \alpha_j + MaxD(s_{m-1}, R_j)\} \qquad (4.3)$$

Uniendo las desigualdades 4.2 y 4.3 tendremos:

$$\begin{aligned} C'(R_j) + \alpha_j \dot{-} C'(R_g) \leq \quad & \text{máx}\{0, \text{mín}\{\alpha_j + MaxD(s_{m-1}, R_j), \\ & MaxR(s_{m-1}, R_j) + \alpha_j - MinR(s_{m-1}, R_g)\}\} \end{aligned}$$

Como esto ocurre para todos los relojes R_j que tienen definida una restricción, tendremos:

$$\begin{aligned} & \max_{\substack{R_j \\ r(R_j) = [\alpha_j, \beta_j]}} \{C'(R_j) + \alpha_j \dot{-} C'(R_g)\} \leq \\ & \max_{\substack{R_j \\ r(R_j) = [\alpha_j, \beta_j]}} (\text{máx}\{0, \text{mín}\{\alpha_j + MaxD(s_{m-1}, R_j), \\ & \qquad MaxR(s_{m-1}, R_j) + \alpha_j - MinR(s_{m-1}, R_g)\}\}) \end{aligned}$$

Si sustituimos en la expresión 4.1 tendremos:

$$\begin{aligned} & Min_\tau(n_{m-1}, C') \leq \\ & \min_{\substack{e \\ n \xrightarrow{e} \\ \lambda_1(e) = \tau}} \{ \max_{\substack{R_j \\ r(R_j) = [\alpha_j, \beta_j]}} \{\text{máx}\{0, \text{mín}\{\alpha_j + MaxD(s_{m-1}, R_j), \\ & \qquad MaxR(s_{m-1}, R_j) + \alpha_j - MinR(s_{m-1}, R_g)\}\}\} \end{aligned}$$

Que es exactamente la definición de Δ_τ, de modo que:

$$Min_\tau(n_{m-1}, C') \leq \Delta_\tau$$

b) Probemos ahora que γ es una cota inferior para $Min(n_{m-1}, C', e)$, es decir, $\gamma \leq Min(n_{m-1}, C', e)$, donde e es el arco que nos permite llegar a n_m a partir de n_{m-1}.

Por la definición de $Min(n, C, e)$ tenemos:

$$Min(n_{m-1}, C', e) = \text{máx}\{C'(R_j) + \alpha_j \dot{-} C'(R_g) | r(R_j) = [\alpha_j, \beta_j]\}$$

Aplicando la hipótesis de inducción para los relojes que tienen definida $r(R_j)$ en el arco e tendremos que se cumple:

$$MinR(s_{m-1}, R_j) \leq C'(R_j) \leq MaxR(s_{m-1}, R_j)$$

Si aplicamos esta desigualdad junto con la definición de $\dot{-}$ a la definición anterior tendremos:

$$\text{máx}\{0, MinR(s_{m-1}, R_j) + \alpha_j - MaxR(s_{m-1}, R_g)\} \leq C'(R_j) + \alpha_j - C'(R_g)$$

Por otro lado, a partir de la definición de discrepancia y de la hipótesis de inducción, de forma inmediata, podemos comprobar que se cumple:

$$\text{máx}\{0, MinD(s_{m-1}, R_j) + \alpha_j\} \leq C'(R_j) + \alpha_j - C'(R_g)$$

Por lo tanto:

$$\begin{aligned} &\text{máx}\{0, max\{MinD(s_{m-1}, R_j) + \alpha_j, \\ &\qquad MinR(s_{m-1}, R_j) + \alpha_j - MaxR(s_{m-1}, R_g)\}\} \leq \\ &C'(R_j) + \alpha_j \dot{-} C'(R_g) \end{aligned}$$

Si ésto se cumple para todo reloj R_j, se cumplirá entonces:

$$\begin{aligned} &\text{máx}\{0, max\{MinD(s_{m-1}, R_j) + \alpha_j, \\ &\qquad MinR(s_{m-1}, R_j) + \alpha_j - MaxR(s_{m-1}, R_g)\}\} \leq \\ &\quad \underset{\substack{R_j \\ r(R_j) = [\alpha_j, \beta_j]}}{\text{máx}} \{C'(R_j) + \alpha_j \dot{-} C'(R_g)\} \end{aligned}$$

Como la parte izquierda de la desigualdad es la definición de γ y la parte derecha es la definición de $Min(n_{m-1}, C', e)$ podemos concluir que

$$\gamma \leq Min(n_{m-1}, C', e)$$

c) A continuación probaremos que δ es una cota superior de $Max(n_{m-1}, C', e)$, donde e es el arco que se va a ejecutar a partir del estado (n_{m-1}, C', t).

Por la definición de $Max(n, C, e)$ tendremos:

$$Max(n_{m-1}, C', e) = \text{mín}\{C'(R_j) + \beta_j - C'(R_g) | r(R_j) = [\alpha_j, \beta_j]\}$$

Aplicando la hipótesis de inducción, para aquellos relojes R_j para los que $r(R_j)$ está definido se cumple:

$$MinR(s_{m-1}, R_j) \leq C'(R_j) \leq MaxR(s_{m-1}, R_j)$$

Aplicando esta desigualdad a la definición anterior tendremos:

$$C'(R_j) + \beta_j - C'(R_g) \leq MaxR(s_{m-1}, R_j) + \beta_j - MinR(s_{m-1}, R_g)\}$$

Por otro lado, aplicando la hipótesis de inducción y considerando la definición de discrepancia, de forma inmediata podemos comprobar que se cumple:

$$C'(R_j) + \beta_j - C'(R_g) \leq MaxD(s_{m-1}, R_j) + \beta_j$$

y por consiguiente:

$$\begin{aligned} C'(R_j) + \beta_j - C'(R_g) \leq \text{mín}\{&MaxD(s_{m-1}, R_j) + \beta_j, \\ &MaxR(s_{m-1}, R_j) + \beta_j - MinR(s_{m-1}, R_g)\}\} \end{aligned}$$

Si ésto se cumple para todo reloj R_j se cumplirá que:

$$\begin{aligned} C'(R_j) + \beta_j - C'(R_g) \leq \min_{\substack{R_j \\ r(R_j) = [\alpha_j, \beta_j]}} \{\text{mín}\{&MaxD(s_{m-1}, R_j) + \beta_j, \\ &MaxR(s_{m-1}, R_j) + \beta_j - MinR(s_{m-1}, R_g)\}\} \end{aligned}$$

Como la parte derecha de la desigualdad es la definición de δ y la parte izquierda es la definición de $Max(n_{m-1}, C', e)$ podemos concluir que:

$$Max(n_{m-1}, C', e) \leq \delta$$

d) Una vez demostrado lo anterior, debemos comprobar que los valores *MinR*, *MaxR*, *MinD*, *MaxD* calculados por el algoritmo para el estado del grafo (n, C, t) cumplen las condiciones (i) y (ii) del enunciado de la proposición.

Para ello distinguiremos dos casos, dependiendo de si los relojes pertenecen al conjunto de sincronización del arco e o no.

- Caso 1 ($R_j \in S$)
  En primer lugar probaremos que los valores *MinR* y *MaxR* cumplen el punto (i). Tras ejecutarse la transición e tendremos que el valor del reloj R_j se ha actualizado y su nuevo valor será el mismo que el de R_g. Ahora, por la función *Adjust* tendremos:

  $$C(R_g) = C'(R_g) + \Delta t$$

por lo que:

$$C(R_j) = C'(R_g) + \Delta t$$

Si la transición ha podido ejecutarse será porque la condición de habilitación de una transición se cumple y por tanto:

$$Min(n_{m-1}, C', e) \leq \Delta t \leq \text{mín}\{Min_\tau(C', e), Max(n_{m-1}, C', e)\}$$

Como hemos demostrado anteriormente se cumple:

$$\gamma \leq Min(n_{m-1}, C', e),\ Max(n_{m-1}, C', e) \leq \delta \text{ y } Min_\tau(n_{m-1}, C') \leq \Delta_\tau$$

Sustituyendo en la desigualdad anterior tendremos:

$$\gamma \leq \Delta t \leq \text{mín}\{\Delta_\tau, \sigma\}$$

Por tanto:

$$C'(R_g) + \gamma \leq C'(R_g) + \Delta t \leq C'(R_g) + \text{mín}\{\Delta_\tau, \sigma\}$$

o lo que es lo mismo:

$$C'(R_g) + \gamma \leq C(R_j) \leq C'(R_g) + \text{mín}\{\Delta_\tau, \sigma\}$$

Como en el nodo n_{m-1} se cumple $MinR(s_{m-1}, R_g) \leq C'(R_g) \leq MaxR(s_{m-1}, R_g)$, tendremos :

$$MinR(s_{m-1}, R_g) + \gamma \leq C(R_j) \leq MaxR(s_{m-1}, R_g) + \text{mín}\{\Delta_\tau, \sigma\}$$

Según la definición de $MinR$ y $MaxR$ se cumple que $MinR(s_{m-1}, R_g) + \gamma = MinR(s, R_j)$ y $MaxR(s_{m-1}, R_g) + \text{mín}\{\Delta_\tau, \sigma\} = MaxR(s, R_j)$. Por tanto:

$$MinR(s, R_j) \leq C(R_j) \leq MaxR(s, R_j)$$

Veamos ahora que los valores $MinD$ y $MaxD$ cumplen el punto (ii). Como $R_j \in S$ se cumplirá que $C(R_j) = C(R_g)$, y por tanto: $C(R_j) - C(R_g) = 0$. Según el algoritmo, $MaxD(s, R_j) = 0$ y $MinD(s, R_j) = 0$. Por tanto, trivialmente:

$$MaxD(s, R_j) \leq C(R_j) - C(R_g) \leq MaxD(s, R_j)$$

- Caso 2: ($R_j \notin S$)
  Comprobaremos en primer lugar que los valores *MinR* y *MaxR* obtenidos por el algoritmo cumplen la condición (i). Si $R_j \notin S$ su valor no se habrá actualizado y por tanto se cumplirá:

$$C(R_j) = C'(R_j)$$

  Por la hipótesis de inducción tenemos:

$$MinR(s_{m-1}, R_j) \leq C'(R_j) \leq MaxR(s_{m-1}, R_j)$$

  Por tanto:

$$MinR(s_{m-1}, R_j) \leq C(R_j) \leq MaxR(s_{m-1}, R_j)$$

  Ahora bien, para este caso se cumple que $MinR(s_{m-1}, R_j) = MinR(s, R_j)$ y $MaxR(s_{m-1}, R_j) = MaxR(s, R_j)$, y por tanto:

$$MinR(s, R_j) \leq C(R_j) \leq MaxR(s, R_j)$$

  Comprobemos ahora que los valores *MinD* y *MaxD* cumplen el apartado (ii).
  Como el reloj R_j no ha sido sincronizado, el valor de su discrepancia se habrá incrementado en tantas unidades como se haya incrementado el valor de R_g. Por lo tanto:

$$D(R_j) = D'(R_j) - \Delta t$$

  Como hemos comprobado en el caso anterior se cumple:

$$\gamma \leq \Delta t \leq \text{mín}\{\Delta_\tau, \delta\}$$

  y también se cumple que

$$MinD(s_{m-1}, R_j) \leq D'(R_j) \leq MaxD(s_{m-1}, R_j)$$

  por lo que tendremos:

$$MinD(s_{m-1}, R_j) - \text{mín}(\Delta_\tau, \delta) \leq D(R_j) \leq MaxD(s_{m-1}, R_j) - \gamma \quad (4.4)$$

  Por otro lado, si partimos de la definición de discrepancia tendremos:

$$D(R_j) = C(R_j) - C(R_g)$$

Como hemos visto anteriormente, para todos los relojes del grafo que no son actualizados se cumple:

$$MinR(s_{m-1}, R_j) \leq C(R_j) \leq MaxR(s_{m-1}, R_j)$$

Sustituyendo en la definición de discrepancia tendremos:

$$MinR(s_{m-1}, R_j) - C(R_g) \leq D(R_j) \leq MaxR(s_{m-1}, R_j) - C(R_g) \quad (4.5)$$

Para el reloj global, que sí ha sido actualizado se cumple:

$$MinR(s_{m-1}, R_g) + \gamma \leq C(R_g) \leq MaxR(s_{m-1}, R_g) + \text{mín}\{\Delta_\tau, \delta\}$$

o lo que es lo mismo:

$$-(MinR(s_{m-1}, R_g)+\gamma) \geq -C(R_g) \geq -(MaxR(s_{m-1}, R_g)+\text{mín}\{\Delta_\tau, \delta\})$$

Sustituyendo en la desigualdad 4.5 tendremos:

$$\begin{aligned} &MinR(s_{m-1}, R_j) - (MaxR(s_{m-1}, R_g) + \text{mín}\{\Delta_\tau, \delta\}) \leq D(R_j) \leq \\ &MaxR(s_{m-1}, R_j) - (MinR(s_{m-1}, R_g) + \gamma) \end{aligned} \quad (4.6)$$

De 4.4 y 4.6 tendremos:

$$\begin{aligned} &\text{máx}\{MinD(s_{m-1}, R_j) - \text{mín}(\Delta_\tau, \delta), \\ &MinR(s_{m-1}, R_j) - MaxR(s_{m-1}, R_g) - \text{mín}\{\Delta_\tau, \delta\}\} \leq D(R_j) \leq \\ &\text{mín}\{MaxR(s_{m-1}, R_j) - MinR(s_{m-1}, R_g) - \gamma, MaxD(s_{m-1}, R_j) - \gamma\} \end{aligned}$$

Lo que concluye la demostración.

□

Ejemplo 24 Consideremos el proceso siguiente:

$$a < 5, 8 > b < 1, 2 >; stop \|_{\{\}} wait(3); d < 1, 2 >; stop$$

Su grafo asociado se muestra en la Figura 4.13. Si consideramos el camino s = 1,2,3,6, obtendremos los valores siguientes:

	R_0	R_1	R_2
$MinR$	4	4	4
$MaxR$	9	5	9
$MinD$	0	4	0
$MaxD$	0	0	0

□

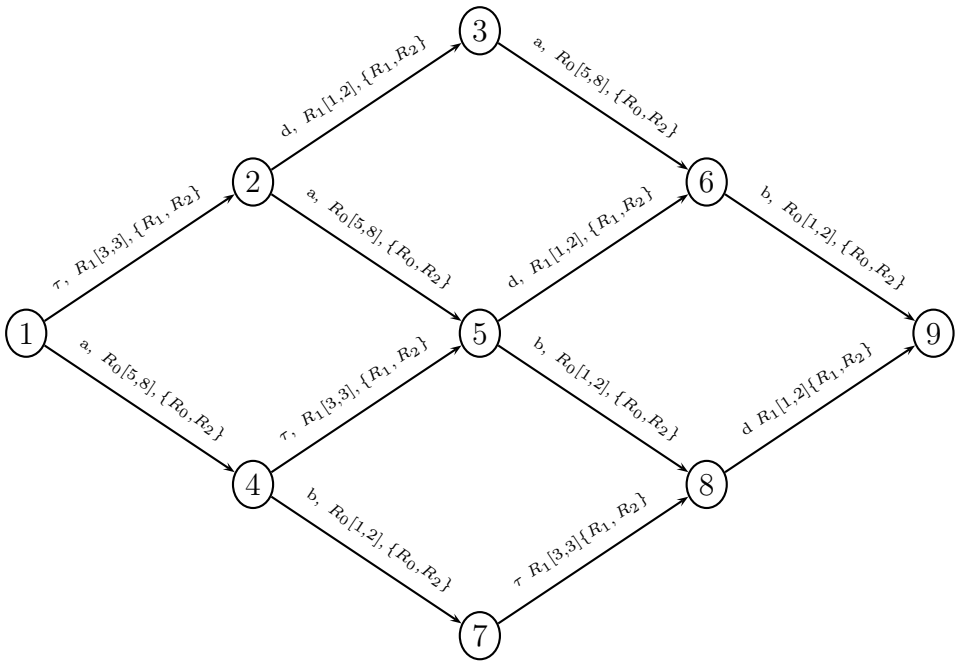


Figura 4.13: Grafo generado para el proceso del ejemplo 24

Algoritmo de reducción

El algoritmo de reducción está basado en un recorrido en anchura de los nodos del grafo, pero en principio sólo será aplicable a grafos acíclicos. Para cada uno de ellos calcularemos las cotas para cada reloj, y para cada camino que permite llegar a ese nodo. Una vez calculadas las cotas podremos determinar si el nodo es alcanzable, y en caso de no ser alcanzable será eliminado, junto con los arcos adyacentes. Si ello implica dejar desconectados otros nodos, estos serán eliminados también.

Para llevar a cabo el recorrido en anchura utilizaremos una cola, en la cual se almacenarán tuplas (n, I, D), donde n será un nodo, I contendrá el conjunto de cotas temporales para cada uno de los relojes de G, y D será el conjunto de discrepancias para los relojes en ese nodo. Tanto las cotas como las discrepancias se calcularán recorriendo el grafo a través de un camino concreto que nos permite alcanzar el nodo n. Si a un nodo llegan varios caminos, se calcularán las cotas para cada uno de los caminos de forma independiente.

Antes de presentar el algoritmo, definiremos el conjunto de nodos adyacentes a uno dado.

Definición 41 (Nodos adyacentes)
Definimos el conjunto de nodos adyacentes a uno dado n, que representamos como $Adyacente(n)$, como el conjunto de los nodos para los cuales existe una transición desde el nodo n.

$$Adyacente(n) = \{m \in N \mid \exists\ e \in E \text{ t.q. } n \xrightarrow{e} m\}$$

□

ALGORITMO DE REDUCCIÓN:

1. Inicialización
   - 1.1 $Q = ColaVacia()$.
   - 1.2 Marcar cada nodo de G como *No_visitado* y cada arco como *no_ejecutable*.
   - 1.3 Insertar (n_0, I_0, D_0) en la cola Q donde $I_0(R_j) = [0, 0]\ \forall R_j \in clocks$.
2. MIENTRAS $\neg EsVacia(Q)$:
   - 2.1 Tomar el elemento (n, I, D) de la cabeza de Q y eliminarlo de la misma.
   - 2.2 Marcar el nodo n como *Visitado*.
   - 2.3 Para cada $m \in Adyacente(n) \wedge \neg Visitado(m)$:
     - 2.3.1 Calcular el nuevo conjunto de cotas $I' = [\alpha, \beta]$ y de discrepancias D' para el nodo m.
     - 2.3.2 SI $\alpha(R_g) \leq \beta(R_g)$ ENTONCES Insertar (m, I', D') en Q y marcar el arco como *ejecutable*.
3. Eliminar todos los nodos etiquetados como *No_visitados* y los arcos etiquetados como *No_ejecutables* del grafo G.

Tanto el algoritmo de cálculo de cotas como el de reducción en principio solamente son aplicables a aquellos grafos que no tengan ciclos, como ocurre por ejemplo con los procesos finitos.

Sin embargo, aunque en general el algoritmo no será aplicable a grafos cíclicos, en algunos casos concretos de procesos recursivos sí que podrá ser utilizado. Concretamente, para procesos de la forma $\mu X.Q$, donde Q no posee nuevas recursiones, el algoritmo podrá ser aplicado, ya que cada vez que lleguemos al nodo inicial tendremos que todos los relojes tienen el mismo valor.

Proposición 5 Sea G un grafo de estados dinámico acíclico y finito. Entonces, el algoritmo de reducción presentado anteriormente termina. Si denotamos por $R'(G)$ el grafo obtenido tras la aplicación del algoritmo anterior, entonces:

$$G \sim R'(G)$$

Demostración: Es inmediata, dado que hemos eliminado aquellos nodos para los cuales no hay ningún camino posible que nos lleve a ellos cumpliendo las restricciones temporales.

□

Proposición 6 Sea $P = \mu X.Q$ un proceso regular donde Q no posee nuevas recursiones, y sea $G[\![P]\!]$ el grafo de estados dinámicos asociado a P.

Entonces el algoritmo de reducción presentado anteriormente termina. Si denotamos por $R'(G)$ al grafo obtenido tras la aplicación del algoritmo anterior, entonces:

$$G \sim R'(G)$$

Demostración: La terminación del algoritmo vendrá garantizada por el hecho de tratarse de un grafo finito, y sólo hemos de aplicar el algoritmo una vez para cada nodo.

La equivalencia es una consecuencia inmediata de la proposición 5 aplicada a $Q(X)$ pues en cada iteración los relojes tienen los mismos valores.

□

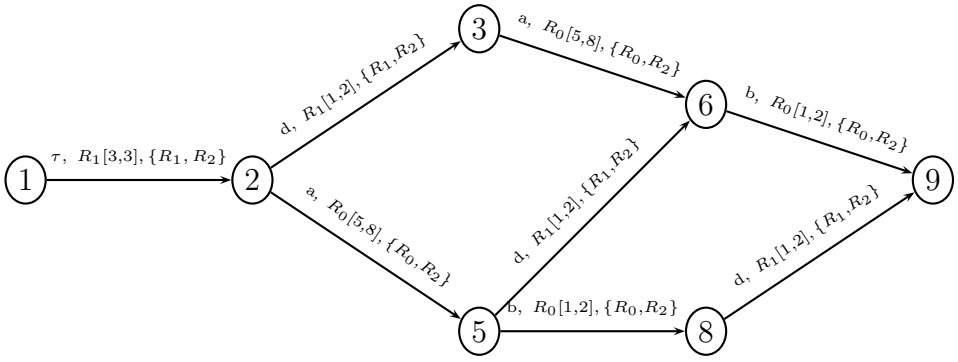
Ejemplo 25 Consideremos de nuevo el grafo del ejemplo 24 (Figura 4.13)

Si calculamos las cotas para los relojes en cada uno de los nodos, y para cada uno de los caminos que lo atraviesan, obtenemos que para el nodo 4 solamente existe un camino que lo atraviesa, por lo que calculamos los valores para los relojes y obtenemos que $MinR(1,4,R_0) = 5$, $MinR(1,4,R_1) = 0$, $MinR(1,4,R_2) = 5$, $MaxR(1,4,R_0) = 3$ $MaxR(1,4,R_1) = 0$, $MaxR(1,4,R_2) = 3$.

Por tanto, este nodo no será alcanzable y será eliminado, así como todos los nodos que únicamente podían ser alcanzados desde él.

Así, en este caso en concreto el grafo reducido será el mostrado en la figura 4.14.

□



$$clocks = \{R_0, R_1, R_2\},\ R_g = R_2$$

Figura 4.14: Grafo reducido correspondiente al ejemplo 24

Otras reducciones

De acuerdo con la relación de equivalencia que hemos definido, basada en trazas temporizadas, podemos encontrar algunos casos particulares que permiten reducir ligeramente el número de nodos y arcos de un grafo. Para ello podemos aplicar las equivalencias siguientes:

Proposición 7 La elección externa es idempotente, es decir, para todo $P \in RTPAL$:

$$G[\![P \Box P]\!] = G[\![P]\!]$$

Demostración: Inmediata.

□

Proposición 8 $\forall\, P, Q \in RTPAL, \forall\, a \in Vis, \forall\, t_1 \in \mathbb{R}_0^+, \forall\, t_2 \in \mathbb{R}_0^+ \cup \{\infty\}$:

$$G[\![a < t1, t2 >; P \ \Box\ a < t1, t2 >; Q]\!] \sim G[\![a < t1, t2 >;\ (\ P \Box Q\ )\ ]\!]$$

Demostración: Inmediata.

□

Proposición 9 $\forall\, P, Q \in RTPAL$:

$$G[\![\tau; P \ \Box\ \tau; Q]\!] \sim G[\![\tau;\ (\ P \Box Q\ )\ ]\!]$$

Demostración: Inmediata.

□

Proposición 10 $\forall\, P, Q \in RTPAL, \forall\, t \in \mathbb{R}_0^+$:

$$G[\![wait(t); P \,\square\, wait(t); Q]\!] \sim G[\![wait(t);\ (\ P \square Q\ )\ ]\!]$$

Demostración: Inmediata.

□

Estas equivalencias abren la puerta a nuevas líneas de trabajo, para estudiar reducciones sintácticas más sofisticadas, que generen grafos más pequeños, y por ello reduzcan el problema de la explosión de estados, ya sea reduciendo el número de nodos o arcos, o incluso reduciendo el número de relojes.

Asimismo, queda abierto el estudio de los grafos cíclicos, en los cuales el algoritmo de reducción presentado no funciona con carácter general. Un primer paso podría ser analizar casos particulares, en los cuales este algoritmo, o una versión modificada del mismo pudiera ser utilizado.

Capítulo 5

Relación con los Autómatas Temporizados

El modelo de grafos de estados dinámicos que hemos presentado tiene muchas similitudes con el modelo de autómatas temporizados. Por ello, hemos considerado oportuno incluir este capítulo, para establecer la relación entre ambos modelos.

En concreto, hemos visto que es posible establecer una traducción de los grafos de estados dinámicos a autómatas temporizados (con invariantes), pero hasta el momento no hemos encontrado una traducción en sentido inverso.

5.1. Traducción de grafos a autómatas

Definición 42
Sea $G = (N,\ E,\ n_0,\ \lambda,\ clocks)$ un grafo de estados dinámico cualquiera.

A partir de G vamos a construir un autómata temporizado A, el cual poseerá el mismo conjunto de nodos N, el mismo conjunto de arcos E y el mismo conjunto de relojes *clocks* que tenía el grafo, mientras que la función *guard* que asigna las guardas a los arcos y la función *inv* que asigna invariantes a los nodos, se definirán a partir de las restricciones que poseen los arcos del grafo.

Formalmente:

$$A = (N,\ n_0,\ E,\ clocks,\ times,\ guard,\ inv)$$

donde

- La función *guard* se define como:

$$guard(e) = \bigwedge_{\substack{x \in clocks \\ \lambda_2(e)(x) = [\alpha_1(e,x),\, \alpha_2(e,x)]}} (\alpha_1(e,x) \leq x \ \wedge \ x \leq \alpha_2(e,x))$$

  Es decir, se realiza una conjunción de todos esos predicados, uno para cada reloj x para el cual exista una restricción en el arco e. Si $\alpha_2(e,x) = \infty$ la segunda componente no aparecerá en la guarda correspondiente.

- La función *times* asignará a cada uno de los arcos del autómata el mismo conjunto de relojes que ese arco tenía asignado en el grafo, pero en este caso serán reseteados en lugar de ser actualizados como ocurría en el grafo.

$$times(e) = \lambda_3(e)$$

- La función *inv* se define como sigue:

$$inv(n) = \begin{cases} True & \text{Si } \nexists\, e \text{ t.q. } n \xrightarrow{e},\ \lambda_1(e) = \tau \\ \bigwedge\limits_{\substack{e,\, n \xrightarrow{e}, \\ \lambda_1(e) = \tau}} \left(\bigvee\limits_{\substack{x \in clocks \\ \lambda_2(e)(x)\, def \\ \lambda_2(e)(x) = [\alpha_1(e,x),\, \alpha_2(e,x)]}} (x \leq \alpha_1(e,x)) \right) & \text{Si } \exists\, e \text{ t.q. } n \xrightarrow{e},\ \lambda_1(e) = \tau \end{cases}$$

  Como puede observarse, se asignan invariantes en aquellos nodos que poseen arcos salientes etiquetados con τ. Para el resto de nodos el invariante asignado es *true*.

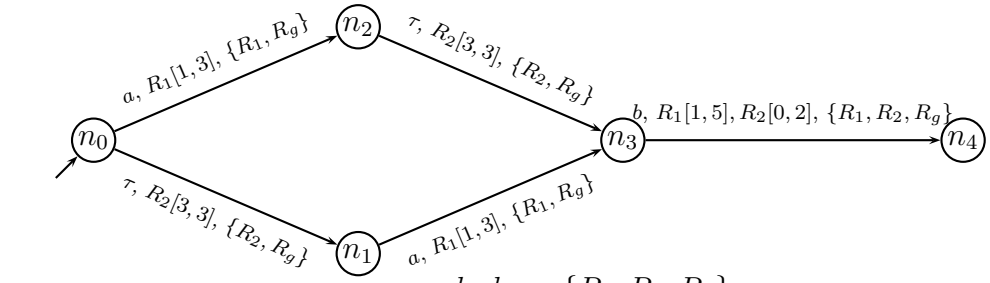
□

Ejemplo 26 Sea G el grafo de estados dinámico mostrado en la Figura 5.1(a). El autómata construido a partir de él es el mostrado en la Figura 5.1(b)

□

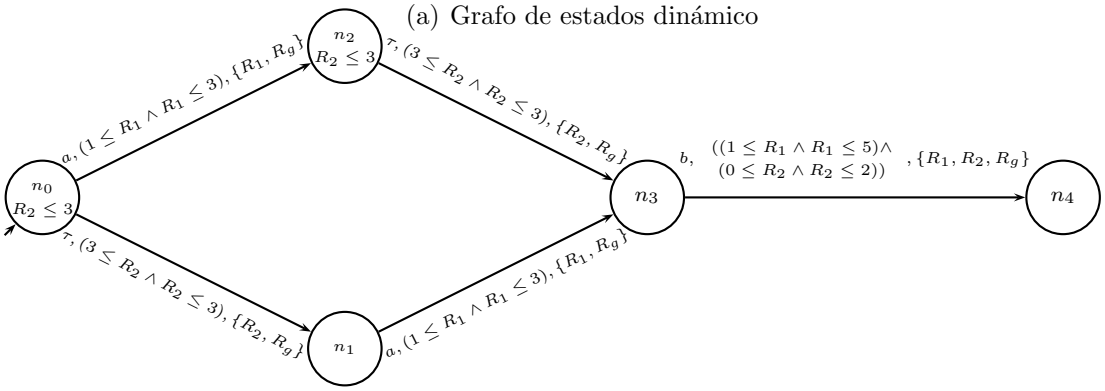
5.2. Equivalencia

Una vez definida la traducción de grafos a autómatas, debemos establecer una noción de equivalencia entre los mismos, que nos permita decidir cuando consideramos que un grafo de estados dinámico y un autómata temporizado tienen la misma conducta.



$clocks = \{R_1, R_2, R_g\}$

(a) Grafo de estados dinámico



$clocks = \{R_1, R_2, R_g\}$

(b) Autómata temporizado obtenido

Figura 5.1: Traducción de un grafo a un autómata

Para poder establecer esta equivalencia debemos relajar la condición de disparo definida en la semántica del autómata (ver Sección 2.3.1). Ésto es debido a que en nuestro modelo de grafos de estados dinámicos no podemos detectar si llegamos a un nodo que posee una transición urgente ya caducada, lo que equivale en el autómata a no poder garantizar que se cumple el invariante en el nodo destino (ver ejemplo 27). Así, la relación de transición para los autómatas temporizados que consideraremos será la definida por las reglas siguientes:

1. $(q, v) \xrightarrow{e} (q',$ reset $times(e)$ in $v)$ si se cumplen las condiciones siguientes:

   1.1 $e = (q, q') \in E$

   1.2 $v \models guard(e)$

   1.3 $v \models inv(q)$

2. $(q, v) \underset{d}{\longmapsto} (q, v + d)$ para un valor real positivo d si se cumple la condición:

   2.1 $\forall d' \leqslant d \;\; v + d' \models inv(q)$

Como puede observarse, para que una transición pueda ser ejecutada solamente exigiremos que los valores de los relojes cumplan el invariante del nodo de partida y no el de llegada de la transición. De este modo, si se llega a un nodo donde no se cumple el invariante, se considerará que el sistema queda bloqueado.

Ejemplo 27 Consideremos el grafo de la Figura 5.2, cuyo autómata asociado es el mostrado en la Figura 5.3.

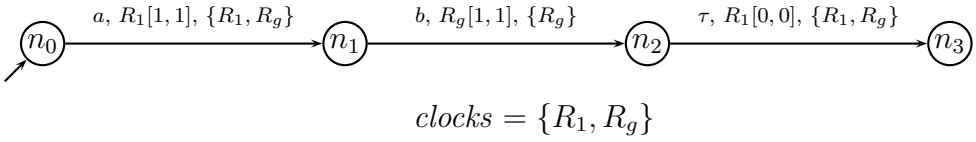


Figura 5.2: Grafo de estados dinámico

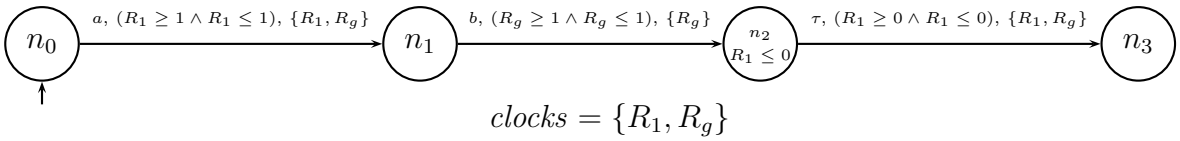


Figura 5.3: Autómata temporizado asociado al grafo de la fig. 5.2

La transición τ no puede ser ejecutada ni en el grafo (ya que no se cumple las restricciones impuestas al arco) ni en el autómata (ya que no se cumple el invariante del nodo de partida).

□

Definición 43 (Bisimulación)
Sean $G = (N_G,\ E_G,\ n_{0,G},\ \lambda_G,\ clocks_G)$ un grafo de estados dinámico y $A = (N_A,\ n_{0,A},\ E_A,\ clocks_A,\ times_A,\ guard_A,\ inv_A)$ un autómata temporizado cualesquiera.

Diremos que G y A son bisimilares, lo que representaremos de la forma $G \sim A$, si y sólo si existe una relación $\rho \subseteq S_G \times S_A$, donde S_G y S_A son respectivamente el conjunto de estados de G y A, tal que:

(I). $(s_{0,G}, s_{0,A}) \in \rho$, siendo $s_{0,G}$ el estado inicial de G y $s_{0,A}$ el estado inicial del autómata.

(II). Si $(s_G, s_A) \in \rho$, $s_G \underset{t}{\longmapsto}_G s'_G$ entonces $\exists\, s'_A$ t.q. $s_A \underset{t}{\longmapsto}_A s'_A$ y $(s'_G, s'_A) \in \rho$

(III). Si $(s_G, s_A) \in \rho$, $s_A \underset{t}{\longmapsto}_A s'_A$ entonces $\exists\, s'_G$ t.q. $s_G \underset{t}{\longmapsto}_G s'_G$ y $(s'_G, s'_A) \in \rho$

(IV). Si $(s_G, s_A) \in \rho$, $s_G \xrightarrow{e}_G s'_G$ entonces $\exists\, s'_A$ t.q. $s_A \xrightarrow{e}_A s'_A$ y $(s'_G, s'_A) \in \rho$

(V). Si $(s_G, s_A) \in \rho$, $s_A \xrightarrow{e}_A s'_A$ entonces $\exists\, s'_G$ t.q. $s_G \xrightarrow{e}_G s'_G$ y $(s'_G, s'_A) \in \rho$

□

Teorema 3

Sea $G = (N,\, E,\, n_0,\, \lambda,\, \mathit{clocks})$ un grafo de estados dinámico y $A[\![G]\!]$ el autómata construido según la definición 42.

Entonces $G \sim A[\![G]\!]$.

Demostración: Para demostrar el teorema anterior definimos la función ρ siguiente, que establece la relación entre ambos:

$$\rho = \{((n, C, t), (n, V))/V(x) = t - C(x),\ x \in \mathit{clocks}\}$$

Una vez definida esta relación comprobemos que cumple las cinco condiciones establecidas en la definición de bisimulación.

(I). Es trivial

(II). Sea (n, C, t) un estado del grafo G y $(n, C, t) \underset{t'}{\longmapsto}_G (n, C, t + t')$ una transición posible en el grafo G a partir de ese estado. Sea (n, V) un estado del autómata construido, de modo que se cumple que $((n, C, t), (n, V)) \in \rho$. Debemos probar que existe un nuevo estado (n, V') en el autómata, que cumple $(n, V) \underset{t'}{\longmapsto}_A (n, V')$ de modo que $V'(x) = V(x) + t',\ \forall\, x \in \mathit{clocks}$, y $((n, C, t + t'), (n, V')) \in \rho$.

Si se ha producido esta transición es porque el grafo permite el paso de t' unidades de tiempo y por tanto tendremos que

$$\Delta t = t + t' - C(R_g) \leq Min_\tau(n, C)$$

Si en esta desigualdad aplicamos la definición de $Min_\tau(n, C)$ tendremos que:

$$t - t' - C(R_g) \leq \min_{\substack{e \\ n \xrightarrow{e} \\ \lambda_1(e) = \tau}} \{Min(n, C, e)\}$$

Aplicando ahora la definición de $Min(n, C, e)$ tendremos que:

$$t+t'-C(R_g) \leq \min_{\substack{e \\ n \xrightarrow{e} \\ \lambda_1(e)=\tau}} \{ \max_{\substack{x \\ \lambda_2(e)(x)def \\ \lambda_2(e)(x)=[\alpha_1(e,x),\alpha_2(ex)]}} \{(C(R_x)+\alpha_1(e,x)) \dot{-} C(R_g)\}\}$$

De aquí podemos deducir que para cualquier arco e tal que $\lambda_1(e) = \tau$ que parta de n se cumple que:

$$t + t' - C(R_g) \leq \max_{\substack{x \\ \lambda_2(e)(x)\ def \\ \lambda_2(e)(x)=[\alpha_1(e,x),\alpha_2(ex)]}} \{(C(R_x) + \alpha_1(e, x)) \dot{-} C(R_g)\}$$

De ahí :

$$\begin{aligned} &\forall\, e,\ \lambda_1(e) = \tau,\ n \xrightarrow{e}, \exists\, x \in \mathit{clocks} \text{ tal que} \\ &\quad 0 \leq t + t' - C(R_g) \leq (C(R_x) + \alpha_1(e, x)) \dot{-} C(R_g) \end{aligned}$$

Dado que $t' > 0$, se cumplirá que $t+t'-C(R_g) > 0$. Ésto nos permite eliminar en ambos lados de la desigualdad el término $C(R_g)$:

$$t + t' \leq C(R_x) + \alpha_1(e, x)$$

o lo que es equivalente:

$$t + t' - C(R_x) \leq \alpha_1(e, x)$$

Como se cumple que $((n, C, t), (n, V)) \in \rho$ tendremos que $V(x) = t - C(R_x)$, y por tanto:

$$t' + V(x) \leq \alpha_1(e, x)$$

Por la definición de ρ tenemos que $V'(x) = V(x) + t'$, por lo que se cumplirá:

$$\forall\, e,\ \lambda_1(e) = \tau,\ n \xrightarrow{e},\ \exists\, x \in \mathit{clocks},\ V'(x) \leq \alpha_1(e, x)$$

de lo que podemos deducir que se cumplirá:

$$\bigwedge_{\substack{e \\ n \xrightarrow{e} \\ \lambda_1(e)=\tau}} (\bigvee_{\substack{x, \\ \lambda_2(e)(x)def. \\ \lambda_2(e)(x)=[\alpha_1(e,x),\alpha_2(ex)]}} (x \leq \alpha_1(e, x)))$$

que es el invariante asociado al nodo n. Por tanto podemos concluir que el autómata A también permite el paso de t' unidades de tiempo.

Tras el paso de las t' unidades de tiempo el estado en el que estará el autómata es (n, V') tal que $V'(x) = V(x) + t'$, y el estado del grafo es $(n, C, t + t')$. Comprobemos que estos dos estados están relacionados, es decir, que se cumple:

$$((n, C, t + t'), (n, V')) \in \rho$$

Para que ésto se cumpla, por la definición de ρ, deberá cumplirse:

$$\forall x \in clocks,\ V'(x) = (t + t') - C(x)$$

En el nuevo estado del autómata (n, V'), dado que con el paso del tiempo no se ha reseteado ninguno de los relojes, los valores de los relojes V' cumplirán:

$$\forall x \in clocks,\ V'(x) = V(x) + t'$$

Dado que se cumple que $((n, C, t), (n, V)) \in \rho$ tendremos que $V(x) = t - C(x)$ y por tanto:

$$\forall x \in clocks,\ V'(x) = t - C(x) + t'$$

o escrito de otra forma:

$$\forall x \in clocks,\ V'(x) = (t + t') - C(x)$$

que es precisamente la definición de la relación de ρ. Por tanto, podemos concluir que $((n, C, t + t'), (n, V')) \in \rho$.

(III). Sea (n, C, t) un estado del grafo y (n, V) un estado del autómata de modo que se cumple que $((n, C, t), (n, V)) \in \rho$. Consideremos un estado del autómata (n, V') tal que $(n, V) \underset{t'}{\longmapsto}_A (n, V')$, $V'(x) = t' + V(x)\ \forall\, x \in clocks$.

En primer lugar comprobaremos que existe un estado $(n, C, t + t')$ alcanzable en el grafo a partir del estado (n, C, t) por el paso de t' unidades de tiempo.

Si el nodo n del autómata posee un invariante distinto de *true*, éste por su definición será de la forma:

$$inv(n) = \bigwedge_{\substack{e \\ n \xrightarrow{e} \\ \lambda_1(e) = \tau}} \Big(\bigvee_{\substack{x \\ \lambda_2(e)(x) def. \\ \lambda_2(e)(x) = [\alpha_1(e,x), \alpha_2(e,x)]}} (x \leq \alpha_1(e, x)) \Big)$$

Si en ese nodo se permite el paso de t' unidades de tiempo es porque se cumple:

$$\forall\, e,\ \lambda_1(e) = \tau,\ n \xrightarrow{e},\ \exists\, x \in clocks,\ V'(x) = t' + V(x) \leq \alpha_1(e, x)$$

De ahí tenemos:

$$t' - t + t + V(x) \leq \alpha_1(e, x)$$

o escrito de otra forma:

$$(t + t') + (V(x) - t) \leq \alpha_1(e, x)$$

Como $V(x) - t = -C(R_x)$ tendremos que $(t + t') - C(R_x) \leq \alpha_1(e, x)$, o lo que es lo mismo:

$$(t + t') \leq C(R_x) + \alpha_1(e, x)$$

Sin que varíe la desigualdad podemos restar a ambos lados de la misma el valor del reloj global con lo que tendremos:

$$(t + t') - C(R_g) \leq C(R_x) + \alpha_1(e, x) - C(R_g)$$

Por tanto:

$$\forall\, e \in E, \lambda_1(e) = \tau,\ (t + t') - C(R_g) \leq \max_x\{(C(x) + \alpha_1(e, x))\ \dot{-}\ C(R_g)\}$$

En consecuencia:

$$t + t' - C(R_g) \leq Min_\tau(n, C)$$

Por lo que el grafo permite pasar t' unidades de tiempo y por tanto:

$$(n, C, t) \underset{t'}{\longmapsto}_G\ (n, C, t + t')$$

Veamos ahora que $((n, C, t + t'), (n, V')) \in \rho$. Para ello, como se cumple $((n, C, t), (n, V)) \in \rho$, tendremos:

$$\forall x \in clocks,\ V(x) = t - C(x)$$

y por tanto:

$$\forall x \in clocks,\ V'(x) = t - C(x) + t'$$

o escrito de otra forma:

$$\forall x \in clocks,\ V'(x) = (t + t') - C(x)$$

que es precisamente la definición de la relación de ρ. Por tanto podemos concluir que $((n, C, t + t'), (n, V')) \in \rho$.

(IV). Consideremos en este caso un estado del grafo (n, C, t) y uno del autómata (n, V) de modo que $((n, C, t), (n, V)) \in \rho$. Además, consideremos que en el grafo se produce la transición $(n, C, t) \xrightarrow{e}_G (m, adjust(C, e, t), t)$. La ejecución de esta transición implica que se cumple la condición $act(s, e)$, es decir, se cumple:

$$Min(n, C, e) \leq t - C(R_g) \leq Max(n, C, e)\ \wedge\ t - C(R_g) \leq Min_\tau(n, C)$$

Para que esa transición pueda ser realizada en el autómata, debemos comprobar que se cumple la guarda de la transición y que los valores de los relojes cumplen el invariante del nodo n.

En primer lugar, comprobaremos que la guarda de la transición a ejecutar en el autómata se cumple, para lo cual partimos de que se cumple:

$$Min(n, C, e) \leq t - C(R_g) \leq Max(n, C, e)$$

Aplicando la definición de $Min(n, C, e)$ y $Max(n, C, e)$ sobre esta desigualdad tendremos que:

$$\begin{array}{l} \max\limits_{\substack{x \\ \lambda_2(e)def\, \lambda_2(e)(x) = [\alpha_1(e,x),\, \alpha_2(e,x)]}} \{\ (C(R_x) + \alpha_1(e,x))\ \dot{-}\ C(R_g)\ \}\ \leq\ t - C(R_g) \wedge \\ t - C(R_g) \leq \min\limits_{\substack{x \\ \lambda_2(e)def \\ \lambda_2(e)(x) = [\alpha_1(e,x),\, \alpha_2(e,x)]}} \{\ (C(R_x) + \alpha_2(e,x)) - C(R_g)\ \} \end{array}$$

Si esta expresión es cierta, se cumplirá que para todo reloj x para el que esté definido $\lambda_2(e)(x)$:

$$(C(R_x) + \alpha_1(e,x))\ \dot{-}\ C(R_g) \leq t - C(R_g) \leq (C(R_x) + \alpha_2(e,x)) - C(R_g)$$

Como se cumple:

$$(C(R_x) + \alpha_1(e,x)) - C(R_g) \leq (C(R_x) + \alpha_1(e,x))\ \dot{-}\ C(R_g)$$

podemos eliminar el término $C(R_g)$ de la expresión anterior con lo que obtenemos:

$$C(R_x) + \alpha_1(e,x) \leq t \leq C(R_x) + \alpha_2(e,x)$$

y restando a toda la expresión el termino $C(R_x)$ nos queda:

$$\alpha_1(e,x) \leq \underbrace{t - C(R_x)}_{V(x)} \leq \alpha_2(e,x)$$

que es precisamente la guarda de la transición a ejecutar.

Tras comprobar que la guarda de la transición se cumple debemos comprobar ahora que los valores de los relojes cumplen el invariante del nodo.

Si la transición puede ser ejecutada en el grafo, se cumplirá:

$$t - C(R_g) \leq Min_\tau(n, C)$$

Si aplicamos la definición de Min_τ tendremos:

$$t - C(R_g) \leq \min_{\substack{e \\ n \xrightarrow{e} \\ \lambda_1(e) = \tau}} \{ \max_{\substack{x \\ \lambda_2(e)\ def \\ \lambda_2(e) = [\alpha_1(e,x), \alpha_2(e,x)]}} \{C(R_x) + \alpha_1(e,x) \dot{-} C(R_g)\}\}$$

Si esto se cumple tendremos:

$$\forall\, e\ \lambda_1(e) = \tau,\ n \xrightarrow{e},\ \exists x\, \lambda_2(x)\ def.,\ t - C(R_g) \leq C(R_x) + \alpha_1(e,x) \dot{-} C(R_g)$$

Como se cumple que $t - C(R_g) \geq 0$ podemos eliminar el término $C(R_g)$ de la expresión anterior, de modo que queda:

$$\forall\, e\ \lambda_1(e) = \tau,\ n \xrightarrow{e},\ \exists x\, \lambda_2(x)\ def.,\ t \leq C(R_x) + \alpha_1(e,x)$$

y reorganizando la expresión:

$$\forall\, e\ \lambda_1(e) = \tau,\ n \xrightarrow{e},\ \exists x\, \lambda_2(x)\ def.,\ t - C(R_x) \leq \alpha_1(e,x)$$

Como para todos los arcos e existe un reloj x que cumple $t - C(R_x) \leq \alpha_1(e,x)$ tendremos que se cumple:

$$\bigwedge_{\substack{e \\ n \xrightarrow{e} \\ \lambda_1(e) = \tau}} (\bigvee_{\substack{x\ t.q. \\ \lambda_2(e)(x) def. \\ \lambda_2(e) = [\alpha_1(e,x), \alpha_2(e,x)]}} x \leq \alpha_1(e,x))$$

Que es precisamente el invariante del nodo origen del arco.

Para finalizar la demostración en este punto debemos comprobar que se cumple que el estado alcanzado en el grafo (m, C', t) y el estado alcanzado en el autómata (m, V') están relacionados, es decir, que se cumple:

$$\forall x \in clocks\ V'(x) = t - C'(x)$$

donde $C' = adjust(C, e, t)$ y $V' = \text{reset } times(e) \text{ in } V$.

Para comprobar esto podemos clasificar los relojes del grafo en dos grupos, dando lugar a los dos casos siguientes:

- $x \in \lambda_3(e)$
  En este caso, los valores de los relojes en el autómata cumplirán que $V'(x) = 0$ y en el grafo $C'(R_x) = t$. Por tanto $V'(x) = t - C'(R_x)$.
- $x \notin \lambda_3(e)$
  Si el reloj no pertenece al conjunto de sincronización su valor no se

habrá modificado en el grafo y en el autómata no se reseteará, por tanto tendremos que $V'(x) = V(x)$ y $C'(R_x) = C(R_x)$. Además, como los nodos de partida en el grafo y en el autómata estaban relacionados se cumple: $V(x) = t - C(R_x)$. Por tanto:

$$V'(x) = t - C'(R_x)$$

y:

$$((m, C', t), (m, V')) \in \rho$$

(v). Consideremos ahora un estado del grafo (n, C, t) y un estado del autómata temporizado (n, V) de modo que $((n, C, t), (n, V)) \in \rho$. Además, consideremos que en el autómata se produce la transición $(n, V) \xrightarrow{e}_A (m, V')$, donde $V' = \text{reset } \mathit{times}(e) \text{ in } V$.

Si esta transición es posible en el autómata es porque la guarda asociada a ese arco se cumple, y por tanto:

$$\forall x \;\; x \in \mathit{guard}(e), \;\; \alpha_1(e, x) \leq V(x) \leq \alpha_2(e, x)$$

Puesto que $((n, C, t), (n, V)) \in \rho$ tenemos que $V(x) = t - C(R_x)$, por lo que sustituyendo en la expresión anterior tendremos:

$$\alpha_1(e, x) \leq t - C(R_x) \leq \alpha_2(e, x)$$

o lo que es lo mismo:

$$C(R_x) + \alpha_1(e, x) \leq t \leq C(R_x) + \alpha_2(e, x)$$

Si en todos los términos de la expresión restamos $C(R_g)$, teniendo en cuenta que $t - C(R_g) \geq 0$, tendremos:

$$(C(R_x) + \alpha_1(e, x)) \;\dot{-}\; C(R_g) \leq t - C(R_g) \leq (C(R_x) + \alpha_2(e, x)) - C(R_g)$$

Esta expresión es cierta para todos los relojes x tal que $x \in \mathit{guard}(e)$, por lo que en particular se cumplirá que:

$$\max_{\substack{x \\ \lambda_2(e)(x) \text{ def.} \\ \lambda_2(e)(x) = [\alpha_1(e,x), \alpha_2(e,x)]}} \{ \; (C(R_x) + \alpha_1(e, x)) \;\dot{-}\; C(R_g) \; \} \;\leq\; t - C(R_g) \;\wedge$$

$$t - C(R_g) \leq \min_{\substack{x \\ \lambda_2(e)(x) \text{ def.} \\ \lambda_2(e)(x) = [\alpha_1(e,x), \alpha_2(e,x)]}} \{ \; (C(R_x) + \alpha_2(e, x)) - C(R_g) \; \}$$

Ésta es la primera parte de la condición que debe cumplirse en el grafo para que pueda ejecutarse la transición.

Por otro lado, para ejecutar la transición en el autómata temporizado debía satisfacerse el invariante en n, el cual era:

$$\bigwedge_{\substack{e \\ n \xrightarrow{e} \\ \lambda_1(e)=\tau}} \left(\bigvee_{\substack{x \ t.q. \\ \lambda_2(e)(x) def. \\ \lambda_2(e)(x)=[\alpha_1(e,x),\alpha_2(e,x)]}} (x \leq \alpha_1(e,x)) \right)$$

Si se cumple esta condición es porque $\forall e \in E,\ n \xrightarrow{e},\ \lambda_1(e) = \tau$, existe $x \in \mathit{clocks}$ tal que $\lambda_2(e)(x)$ está definido, siendo $\lambda_2(e)(x) = [\alpha_1(e,x), \alpha_2(e,x)]$ de modo que $V(x) \leq \alpha_1(e,x)$. De ahí:

$$t - C(R_x) \leq \alpha_1(e,x)$$

Sumando $C(R_g)$ a ambos términos y reorganizando:

$$t - C(R_g) \leq \alpha_1(e,x) + c(R_x) - C(R_g) \leq \max_{\substack{x \in \mathit{clocks} \\ \lambda_2(e)(x) def \\ \lambda_2(e)(x)=[\alpha_1(e,x),\alpha_2(e,x)]}} \{(C(R_x) + \alpha_1(e,x)) \dot{-} C(R_g)\}$$

De ahí:

$$t - C(R_g) \leq \mathit{Min}_\tau(n, C)$$

lo que prueba la segunda parte de la condición de habilitación de la transición. Debemos comprobar que el estado alcanzable en el grafo y en el autómata están relacionados, es decir:

$$((m, C', t), (m, V')) \in \rho$$

con

$$V'(x) = t - C'(R_x)\ \forall\, x$$

donde $C' = \mathit{adjust}(C, e, t)$ y $V' = \text{reset } \mathit{times}(e) \text{ in } V$.

Si $x \in \mathit{times}(e)$, su valor habrá sido reseteado con lo que $V'(x) = 0$, y en el grafo se habrá actualizado, con lo que $C'(R_x) = t$. Por lo tanto:

$$V'(x) = t - C'(R_x)$$

Si $x \notin \mathit{times}(e)$ en el autómata el valor del reloj no habrá sido reseteado por lo que se cumplirá $V'(x) = V(x)$, y en el grafo no habrá sido actualizado, por lo que $C'(R_x) = C(R_x)$.

Como los nodos de partida están relacionados se cumplirá:

$$\forall\, x:\ V(x) = t - C(R_x)$$

Sustituyendo las igualdades anteriores tendremos:

$$\forall\, x:\ V'(x) = t - C'(R_x)$$

Por tanto podemos concluir que:

$$((m, C', t), (m, V')) \in \rho$$

□

Con respecto a la traducción inversa, es decir, obtener un grafo de estados dinámico a partir de un autómata temporizado, nos encontramos con diferentes problemas relacionados fundamentalmente con la traducción de los invariantes, y la dificultad de capturarlos mediante acciones urgentes, lo que ha hecho que no hayamos encontrado una traducción satisfactoria. Esta situación nos hace pensar que los grafos de estados dinámicos son computacionalmente algo más débiles que los autómatas temporizados, aunque ésto tampoco hemos podido demostrarlo formalmente.

Capítulo 6

Traducción del Álgebra a Redes de Petri

En este capítulo definiremos una traducción de los procesos temporizados de TPAL a un modelo temporizado de Redes de Petri, concretamente a Redes de Petri con arcos temporizados etiquetadas y marcadas (MLTAPN).

Con esta traducción lo que se pretende conseguir es unir la facilidad de especificación que presentan las álgebras de procesos con las ventajas que plantean las redes de Petri, entre las que podemos destacar las siguientes: el carácter gráfico de las redes de Petri que hace de éstas un modelo legible y fácil de entender, la existencia de una cierta variedad de métodos de análisis y sobre todo la disponibilidad de diversas herramientas que nos permiten el manejo y análisis de las redes de forma automatizada.

Consideraremos en primer lugar el caso finito, definiendo formalmente la red asociada a cada uno de los operadores del lenguaje, a excepción de la recursión.

La generalización a procesos infinitos requiere considerar de nuevo una restricción sintáctica, pues ésta sólo funciona, en principio, para los llamados términos regulares. No obstante, veremos cómo podemos explotar las características del modelo de redes utilizado para extender esta construcción a un cierto tipo de términos no regulares.

6.1. Caso Finito

Denotaremos por *fTPAL* al conjunto de términos de *TPAL* en los cuales no aparece el operador recursión. A continuación se define la red de Petri con arcos temporizados etiquetada (MLTAPN) asociada a cada uno de los operadores del lenguaje *fTPAL*.

6.1.1. Stop

La MLTAPN asociada con el proceso *stop* consta de un único lugar, el cual estará marcado con un único token, cuya edad inicial es cero.

$$N[\![stop]\!] = (\{s_0\}, \emptyset, \emptyset, \emptyset, \emptyset)$$
$$M_0(s_0) = \{0\}$$

6.1.2. Prefijo temporizado

Sea $P = a < x_1, x_2 >; P_1$ un proceso perteneciente a *fTPAL*, donde la MLTAPN asociada al proceso P_1 es $N_1 = (S_1, T_1, F_1, times_1, \lambda_1, M_{01})$.

A partir de esta red construimos la red asociada al proceso P añadiendo un nuevo lugar, y una nueva transición. Esta transición tendrá como lugar precondición el nuevo lugar introducido, y como lugares postcondición todos los lugares iniciales de N_1 (Definición 20).

Formalmente, definimos la nueva red como:

$$N = (S, T, F, times, \lambda, M_0)$$

donde:

$$S = \{s_0\} \cup S_1$$
$$T = \{t_0\} \cup T_1$$
$$F = F_1 \cup \{(s_0, t_0)\} \cup \{(t_0, s_{0,1}) \mid s_{0,1} \in Init(N_1)\}$$
$$times(s,t) = \begin{cases} times_1(s,t) & \text{si } (s,t) \in F_1 \\ (x_1, x_2) & \text{si } (s,t) = (s_0, t_0) \end{cases}$$
$$\lambda(t) = \begin{cases} \lambda_1(t) & \text{si } t \in T_1 \\ a & \text{si } t = t_0 \end{cases}$$

$$M_0(s) = \begin{cases} \{0\} & \text{si } s = s_0 \\ \emptyset & \text{en caso contrario} \end{cases}$$

Como puede observarse, el tamaño de la nueva red N será el tamaño de N_1 incrementado en una transición, un lugar y un cierto número de arcos (tantos como lugares haya en $Init(N_1)$).

6.1.3. Prefijo de acción urgente

La definición de la red asociada al proceso $P = \tau; P_1$ es similar a la realizada para el prefijo temporizado, con la diferencia de que la etiqueta de la nueva transición introducida será τ, y la etiqueta para el nuevo arco introducido (s_0, t_0) es $[0, 0]$.

Estas variaciones en la construcción hacen que varíe la definición de las funciones de etiquetado, que para este operador serán las siguientes:

$$times(s,t) = \begin{cases} times_1(s,t) & \text{si } (s,t) \in F_1 \\ (0,0) & \text{si } (s,t) = (s_0, t_0) \end{cases}$$

$$\lambda(t) = \begin{cases} \lambda_1(t) & \text{si } t \in T_1 \\ \tau & \text{si } t = t_0 \end{cases}$$

6.1.4. Operador wait

Para el proceso $P = wait(t); P_1$ la construcción de la MLTAPN asociada es similar a la del prefijo temporizado; pero, al igual que ocurría con el operador de prefijo de acción urgente, variarán las funciones de etiquetado. Para este operador, éstas serán:

$$times(s,t) = \begin{cases} times_1(s,t) & \text{si } (s,t) \in F_1 \\ (t,t) & \text{si } (s,t) = (s_0, t_0) \end{cases}$$

$$\lambda(t) = \begin{cases} \lambda_1(t) & \text{si } t \in T_1 \\ \tau & \text{si } t = t_0 \end{cases}$$

Ejemplo 28 Consideremos el siguiente proceso perteneciente a *fTPAL*:

$$P = \tau; a < 4, 5 >; wait(3); b < 0, 3 >; stop$$

La MLTAPN que obtendríamos para este proceso es la mostrada en la Figura 6.1. □

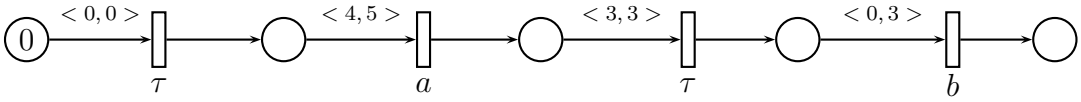


Figura 6.1: MLTAPN para el proceso del ejemplo 28

6.1.5. Elección externa

Cuando deseamos construir la MLTAPN asociada al proceso $P = P_1 \Box P_2$ debemos tener en cuenta la forma de los procesos P_1 y P_2, y por tanto de las MLTAPN asociadas a los mismos, N_1 y N_2. Concretamente nos fijaremos en el número de lugares marcados inicialmente que poseen cada una de las redes. Así, en el caso de que ambas redes posean sólo un lugar marcado inicialmente, la nueva red será muy simple, al igual que el procedimiento para obtenerla, el cual consistirá en añadir solamente un nuevo lugar, eliminando los lugares iniciales de ambas redes, y conectando el nuevo lugar introducido con las transiciones postcondición de los lugares eliminados.

Si, por el contrario, alguna de las redes posee más de un lugar marcado inicialmente, la construcción de la red asociada al proceso P será algo más sofisticada, siendo necesario añadir varios lugares, uno por cada par de transiciones iniciales de N_1 y N_2.

Sean $P_1, P_2 \in fTPAL$ dos procesos cuyas redes asociadas son, respectivamente, $N_1 = (S_1, T_1, F_1, times_1, \lambda_1, M_{01})$ y $N_2 = (S_2, T_2, F_2, times_2, \lambda_2, M_{02})$.

La red para el proceso $P = P_1 \Box P_2$ se definirá como sigue:

$$N[\![P_1 \Box P_2]\!] = \begin{cases} N_1 \,\Box_1\, N_2 & \text{si } |Init(N1)| = 1 \wedge |Init(N_2)| = 1 \\ \\ N_1 \,\Box_2\, N_2 & \text{en caso contrario} \end{cases}$$

donde $\Box_1$ y $\Box_2$ son operadores sobre MLTAPN que definimos a continuación, e *Init* se define como en la definición 20.

Definición 44 (Operador $\Box_1$)
Sean $N_1 = (S_1, T_1, F_1, times_1, \lambda_1, M_{01})$ y $N_2 = (S_2, T_2, F_2, times_2, \lambda_2, M_{02})$ dos MLTAPN que cumplen que $|Init(N1)| = 1$ y $|Init(N_2)| = 1$.

Definimos la red $N = N_1 \,\Box_1\, N_2$ como:

$$N_1 \,\square_1\, N_2 = (S, T, F, times, \lambda, M_0)$$

donde:

$$\begin{aligned}
&S = \{s_0\} \cup S_1 \cup S_2 \setminus (Init(N_1) \cup Init(N_2)) \\
&T = T_1 \cup T_2 \\
&F = \{(s_0, t_i) \mid (s_i, t_i) \in F_i, s_i \in Init(N_i), i = 1, 2\} \cup (F_1 \cup F_2) \\
&\qquad \setminus \{(s_i, t_i) \mid (s_i, t_i) \in F_i, s_i \in Init(N_i), i = 1, 2\} \\
&times(s,t) = \begin{cases} times_i(s,t) & \text{si } (s,t) \in F_i \\ times_i(s_i,t) & \text{si } s = s_0, s_i \in Init(N_i), (s_i,t) \in F_i \end{cases}
\end{aligned}$$

$$\begin{aligned}
&\lambda(t) = \lambda_i(t), \text{ donde } t \in T_i \\
&M(s) = \begin{cases} \emptyset & \text{si } s \neq s_0 \\ \{0\} & \text{si } s = s_0 \end{cases}
\end{aligned}$$

□

El tamaño de la red generada por el operador $\square_1$ será esencialmente la suma de los tamaños de las redes argumentos, ya que solamente se elimina un lugar.

Definición 45 (Operador $\square_2$)
Sean $N_1 = (S_1, T_1, F_1, times_1, \lambda_1, M_{01})$ y $N_2 = (S_2, T_2, F_2, times_2, \lambda_2, M_{02})$ dos MLTAPN que cumplen que $|Init(N_1)| > 1$ ó $|Init(N_2)| > 1$.

Definimos la red $N = N_1 \,\square_2\, N_2$ como:

$$N_1 \,\square_2\, N_2 = (S, T, F, times, \lambda, M_0)$$

donde:

$$\begin{aligned}
&S = \{(t_{i1}, t_{i2}) \mid t_{i1} \in Init(N_1)^\bullet, t_{i2} \in Init(N_2)^\bullet\} \cup S_1 \cup S_2 \\
&T = T_1 \cup T_2 \\
&F = F_1 \cup F_2 \cup \{((t_{i1}, t_{i2}), t_{i1}), ((t_{i1}, t_{i2}), t_{i2}) \mid (t_{i1}, t_{i2}) \in S\} \\
&times(s,t) = \begin{cases} times_i(s,t) & \text{si } (s,t) \in F_i \\ (0, MW(P_1 \square P_2)) & \text{si } s = (t_{i1}, t_{i2}) \end{cases} \\
&\lambda(t) = \lambda_i(t), \text{ donde } t \in T_i \\
&M(s) = \begin{cases} \{0\} & \text{si } s = (t_{i1}, t_{i2}) \vee s \in Init(N_i) \\ \emptyset & \text{en caso contrario} \end{cases}
\end{aligned}$$

La función $MW(P)$ nos proporciona una cota superior para la edad máxima que puede tener un token no muerto en un lugar inicial, pudiendo ser esta edad infinito. La definición de esta función es la siguiente:

$$
\begin{aligned}
&MW(stop) = 0\\
&MW(a < x_1, x_2 >; P) = x_2\\
&MW(\tau; P) = 0\\
&MW(wait(x); P) = x\\
&MW(P_1 \Box P_2) = \text{máx}(MW(P_1), MW(P_2))\\
&MW(P \backslash a) = MW(P)\\
&MW(P_1 \|_A P_2) = \text{máx}(MW(P_1), MW(P_2))
\end{aligned}
$$

□

El tamaño de la red obtenida por aplicación del operador $\Box_2$ será mayor que la suma de los tamaños de las redes argumento, ya que se introducirán nuevos lugares y arcos. Concretamente, el número de nuevos lugares que introduciremos será:

$$|Init(N_1)^\bullet| \times |Init(N_2)^\bullet|$$

mientras que el número de nuevos arcos introducidos será:

$$2 \times |Init(N_1)^\bullet| \times |Init(N_2)^\bullet|$$

Ejemplo 29 Consideremos los procesos $P_1 = a < 1, 2 >; \tau; stop \,\Box\, b < 0, 3 >; stop$ y $P_2 = c < 0, 0 >; stop \,\|_\emptyset\, d < 1, 4 >; stop$. Las MLTAPN correspondientes a ambos procesos se muestran en las Figuras 6.2(a) y 6.2(b); mientras que la red para el proceso $P_1 \Box P_2$ se muestra en la Figura 6.2(c).

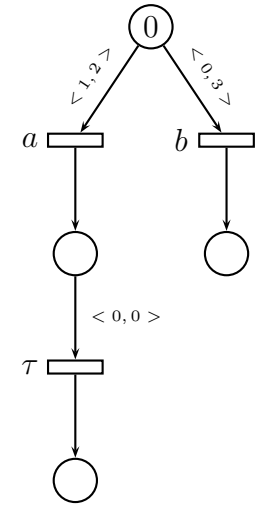
□

6.1.6. Ocultamiento

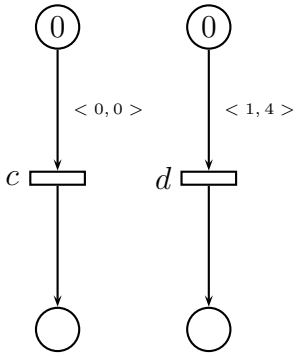
Sea P un proceso perteneciente a *fTPAL*, cuya MLTAPN asociada es:

$$N = (S, T, F, times, \lambda, M_0)$$

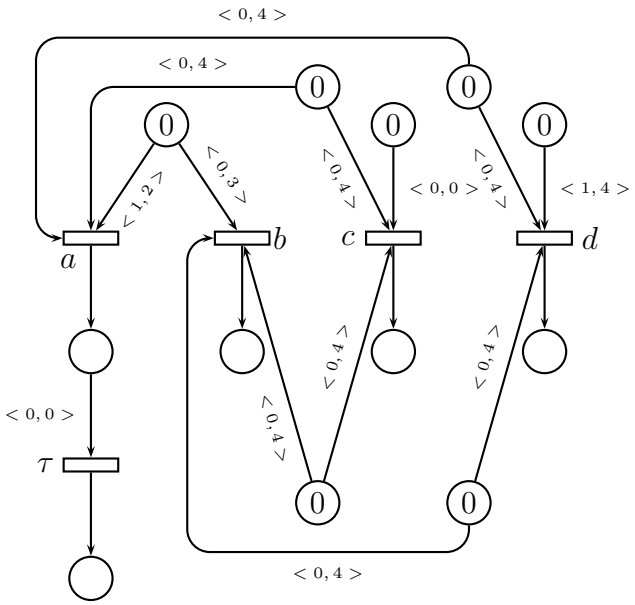
Para construir la red asociada al proceso $P_1 = P \backslash a$ solamente necesitamos cambiar las etiquetas de las transiciones etiquetadas con a por τ.



(a) Red de N_1



(b) Red de N_2



(c) Red de $N_1 \square N_2$

Figura 6.2: MLTAPN para los procesos del ejemplo 29

6.1.7. Composición Paralela

Sean $P_1,\ P_2$ dos procesos pertenecientes a $fTPAL$, cuyas redes asociadas son $N_1 = (S_1, T_1, F_1, times_1, \lambda_1, M_{01})$ y $N_2 = (S_2, T_2, F_2, times_2, \lambda_2, M_{02})$, respectivamente.

La red asociada al proceso $P = P_1 \|_A P_2$ se define como sigue:

$$N = (S,\ T,\ F,\ times,\ \lambda,\ M_0)$$

donde

$$
\begin{aligned}
S &= S_1 \cup S_2 \\
T &= ((T_1 \cup T_2) \setminus \{t_i \in T_i \mid \lambda_i(t_i) \in A, i = 1,2\}) \cup \\
&\quad \{(t_1,t_2) \mid t_1 \in T_1,\ t_2 \in T_2,\ \lambda_1(t_1) = \lambda_2(t_2) \in A\} \\
F &= (F_1 \cup F_2 \mid_{(S\times T)\cup(T\times S)} \cup \{(s_1,(t_1,t_2)) \mid (s_1,t_1) \in F_1\} \cup \\
&\quad \{(s_2,(t_1,t_2)) \mid (s_2,t_2) \in F_2\} \cup \{((t_1,t_2),s_1) \mid (t_1,s_1) \in F_1\} \cup \\
&\quad \{((t_1,t_2),s_2) \mid (t_2,s_2) \in F_2\}
\end{aligned}
$$

$$
times(s,t) = \begin{cases} times_i(s,t) & \text{si } (s,t) \in F_i \\ times_1(s_1,t_1) & \text{si } t = (t_1,t_2) \wedge (s_1,t_1) \in F_1 \\ times_2(s_2,t_2) & \text{si } t = (t_1,t_2) \wedge (s_2,t_2) \in F_2 \end{cases}
$$

$$
\lambda(t) = \begin{cases} \lambda_i(t) & \text{si } t \in T_i \\ \lambda_1(t_1) & \text{si } t = (t_1,t_2) \end{cases}
$$

$$
M_0(s) = M_{0i}(s) \text{ donde } s \in S_i
$$

Como puede observarse en la definición anterior, lo que hacemos es crear una nueva transición por cada par de transiciones (t_1, t_2) cuya etiqueta sea una acción perteneciente al conjunto A. Esta nueva transición sustituye al par que la genera, siendo su conjunto de precondiciones (postcondiciones) la unión de las precondiciones (postcondiciones) de las dos transiciones involucradas.

El tamaño de la nueva red N será posiblemente algo menor a la suma de los tamaños de las redes N_1 y N_2, dependiendo del número de sincronizaciones realizadas.

Ejemplo 30 Sean los procesos $P_1 = a < 2,4 >; b < 1,3 >; stop$, cuya red asociada se muestra en la Figura 6.3(a) y $P_2 = b < 0,4 >; c < 1,2 >; stop$, cuya red se muestra en la Figura 6.3(b).

La MLTAPN que obtendremos para el proceso $P_1 \|_{\{b\}} P_2$ será la mostrada en la Figura 6.3(c).

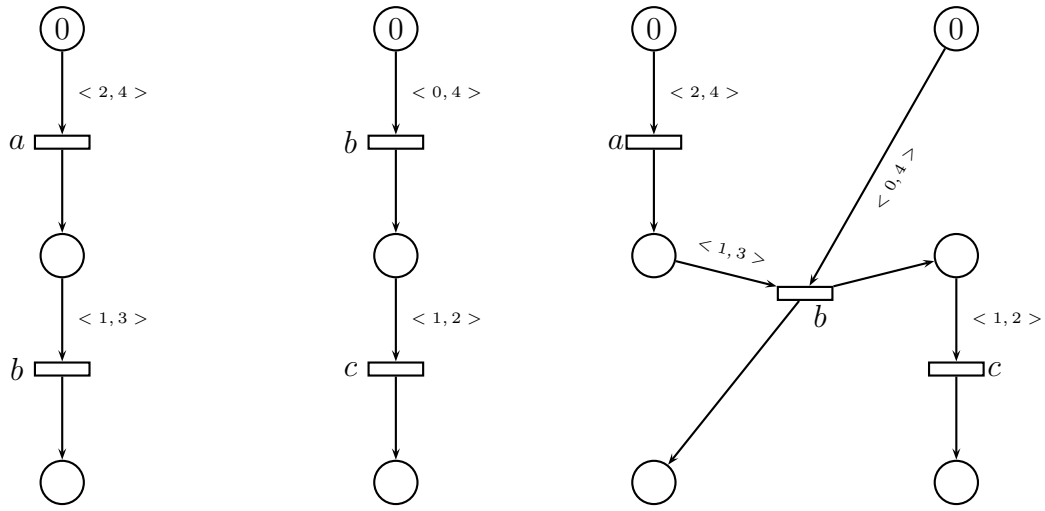
□

Lema 2 Para todo proceso $P \in fTPAL$, la red $N[\![P]\!]$ es segura.

Demostración: Es inmediata.

□

Teorema 4 Sea $P \in fTPAL$ un proceso finito y (N, M_0) la MLTAPN asociada al mismo, construida de acuerdo con la definición anterior.



(a) $N[\![P_1]\!]$ (b) $N[\![P_2]\!]$ (c) $N[\![P_1 \|_{\{b\}} P_2]\!]$

Figura 6.3: MLTAPN para los procesos del ejemplo 30

El sistema de transiciones etiquetado asociado con al proceso P, $lts(P)$, y el grafo de alcanzabilidad de la MLTAPN (N, M_0) son *bisimilares*, es decir, existe una relación:

$$\rho \subseteq fTPAL \times \mathcal{M}(N)$$

tal que:

1. $(P, M_0) \in \rho$
2. $\forall (Q, M) \in \rho$:
   - 2.1 Si $Q \underset{x}{\longmapsto} Q'$ entonces (N, M) puede envejecer x unidades de tiempo y $(Q', age(M, x)) \in \rho$.
   - 2.2 Si $Q \xrightarrow{a} Q'$ entonces $\exists\, t \in T,\ \lambda(t) = a$ tal que $M[t\rangle M'$ con $(Q', M') \in \rho$.
   - 2.3 Si (N, M) puede envejecer x unidades de tiempo, entonces existe una transición $Q \underset{x}{\longmapsto} Q'$ con $(Q', age(M, x)) \in \rho$.
   - 2.4 Si $M[t\rangle M'$, entonces existe una transición $Q \xrightarrow{\lambda(t)} Q'$ con $(Q', M') \in \rho$.

Demostración:

Para realizar esta demostración procederemos por inducción estructural.

- Caso Base

  El caso base es el proceso *stop*. En este caso la demostración es inmediata.

- Caso General

  En el caso general distinguiremos varios casos dependiendo del operador.

  - Prefijo Temporizado: $P = a < t_1, t_2 >; P_1$

    Aplicando la hipótesis de inducción sobre P_1 obtenemos la función ρ_1 que establece la equivalencia entre este proceso y su MLTAPN asociada.

    Para construir la función ρ tendremos que añadir algunos elementos a la función ρ_1, los cuales corresponden a la situación inicial. Así, la función ρ será:

    $$\begin{aligned}\rho = \quad & \{(a < t_1, t_2 >; P_1, M_0)\} \cup \\ & \{(a < t_1 \dot{-} t, x_2 - t >; P_1, age(M_0, t)) \mid t \leq t_2 \} \cup \\ & \{(stop,\ age(M_0, t)) \mid t > t_2 \} \cup \\ & \{(Q, M) \mid M|_{Init(N[\![P]\!])} = \emptyset, (Q, M|_{S_1}) \in \rho_1\}\end{aligned}$$

    donde S_1 es el conjunto de lugares de la red $N[\![P_1]\!]$.

    Una vez definida la relación ρ comprobaremos que con esta definición se cumplen las condiciones establecidas por el teorema.

    1. Inmediata.

    2.1 Podemos distinguir dos casos dependiendo de la cantidad de tiempo x que transcurra.

    - Caso a ($x \leq t_2$)

      Tras el paso de x unidades de tiempo, el proceso P pasará a comportarse como el proceso $P' = a < t_1 \dot{-} x, t_2 - x >; P_1$. $N[\![P]\!]$ con el marcado M_0 permitirá el paso de las x unidades de tiempo, ya que no existirá ninguna transición etiquetada con τ que sea postcondición del lugar inicial de la red. Tras el paso de esa cantidad de tiempo el marcado alcanzado en la red será $age(M_0, x)$. Por la definición de ρ tenemos:

      $$(a < t_1 \dot{-} x, t_2 - x >; P_1, age(M_0, x)) \in \rho$$

    - Caso b ($x > t_2$)

      Tras el paso de x unidades de tiempo, el proceso P pasará a comportarse como el proceso *stop*, que únicamente permitirá el paso del tiempo.

$N[\![P]\!]$ con el marcado M_0 permitirá el paso de las x unidades de tiempo, ya que no existirá ninguna transición etiquetada con τ que sea postcondición del lugar inicial de la red. Tras el paso de esa cantidad de tiempo el marcado alcanzado en la red será $age(M_0, x)$, el cual habrá convertido en muerto el token que hay en el lugar inicial, con lo cual la red no podrá evolucionar y lo único que permitirá será el paso del tiempo.
Por la definición de ρ tenemos:

$$(stop, age(M_0, x)) \in \rho$$

2.2 La única posibilidad de que el proceso P pueda realizar la acción a es que $t_1 = 0$. Tras ejecutar esta acción el proceso P pasará a comportarse como el proceso P_1.
$N[\![P]\!]$ tendrá una transición t tal que $\lambda(t) = a$, un arco e que une el lugar inicial con la transición t tal que $times(s_0, t) = (0, t_2)$, y el marcado de esta red solamente poseerá un token en el lugar inicial cuya edad será 0. Por tanto esa transición estará habilitada para ser ejecutada, y tras su ejecución el marcado alcanzado M' cumplirá que $M'|_{Init(N[\![P]\!])} = \emptyset$ y $M'|_{S_1} = M_{01}$.
Por la hipótesis de inducción se cumple $(P_1, M_{01}) \in \rho_1$ y por tanto

$$(P_1, M') \in \rho$$

2.3 $N[\![P]\!]$ con el marcado M_0 permite el paso de las x unidades de tiempo tras lo cual alcanza el marcado $age(M_0, x)$.
Si $x \leq t_2$ el proceso P también permitirá el paso de x unidades de tiempo, tras lo cual pasará a comportarse como el proceso $P' = a < t_1 \dot{-} x, t_2 - x >; P_1$ y por la definición de ρ tenemos:

$$(a < t_1 \dot{-} x, t_2 - x >; P_1, age(M_0, x)) \in \rho$$

Si $x > t_2$ el proceso P también permite el paso de esa cantidad de tiempo tras lo cual pasará a comportarse como $P' = stop$ y por la definición de ρ

$$(stop, age(M_0, x)) \in \rho$$

2.4 $N[\![P]\!]$ tendrá una transición t tal que $\lambda(t) = a$, que se encuentra habilitada bajo el marcado M_0, es decir existe un arco (s_0, t) tal

que $times(s_0, t) = 0$. Tras disparar esta transición se alcanzará el marcado M' que cumplirá que $M'|_{Init(N[\![P]\!])} = \emptyset$ y $M'|_{S_1} = M_{01}$. Como $P = a < 0, t_2 >; P_1$, éste podrá ejecutar la acción a, tras lo cual pasará a comportarse como P_1, cumpliéndose por la hipótesis de inducción que $(P_1, M_0 1) \in \rho$ y por tanto:

$$(P_1, M') \in \rho$$

- Prefijo de acción urgente: $P = \tau; P_1$
  Aplicando la hipótesis de inducción sobre P_1 obtenemos la función ρ_1 que establece la equivalencia entre este proceso y su MLTAPN asociada.

  Para obtener la nueva función ρ debemos añadir algunos elementos nuevos a la función ρ_1, los cuales corresponden a la situación inicial. Así, la definición de la función ρ es la siguiente::

$$\rho = \{(\tau; P, M_0)\} \cup \{(Q, M) \mid M|_{Init(N[\![P]\!])} = \emptyset, (Q, M|_{S_1}) \in \rho_1\}$$

  donde S_1 es el conjunto de lugares de la red $N[\![P_1]\!]$. Una vez definida la relación ρ demostremos que se cumplen cada uno de los puntos establecidos por el teorema.

  1. Inmediata.

     2.1 La única acción que puede realizar el proceso P es una τ, que debe ser realizada en tiempo 0, por lo que no permitirá el paso del tiempo.
     $N[\![P]\!]$ tiene una transición etiquetada con τ y el arco que une el lugar inicial con esa transición está etiquetado con $(0, 0)$. Además el marcado inicial M_0 tiene un único token con edad 0 en el lugar inicial por lo que la transición estará habilitada para su disparo y debe ser ejecutada. Por lo tanto $N[\![P]\!]$ con el marcado M_0 tampoco permite el paso del tiempo.

     2.2 La única acción que puede ejecutar el proceso P es la acción τ y la deberá realizar en el instante 0. Tras ejecutar esa acción alcanzaremos el proceso P_1.
     En la MLTAPN asociada, tenemos una transición t cuya etiqueta es $\lambda(t) = \tau$, la cual tiene un único lugar precondición que es el lugar inicial de la red. Además, el marcado de este lugar tendrá un único token x con edad 0, es decir $M(s_0)(x) = 0$. También ten-

dremos que $times(s_0, t) = (0, 0)$, por lo que la transición está habilitada y debe ser ejecutada en el instante 0. Tras ejecutarse esa transición se alcanzará un marcado M' donde $M'|_{Init([\![P]\!])} = \emptyset$.
Por la hipótesis de inducción tenemos que $(P_1, M_{01}) \in \rho_1$, y por la construcción de ρ tenemos que $(P_1, M') \in \rho$.

2.3 $N[\![P]\!]$ tiene una transición etiquetada con τ y el arco que une el lugar inicial con esa transición está etiquetado con $(0, 0)$. Además el marcado inicial M_0 tiene un único token con edad 0 en el lugar inicial por lo que la transición estará habilitada para su disparo y debe ser ejecutada. Por lo tanto $N[\![P]\!]$ con el marcado M_0 no permite el paso del tiempo.
Como $P = \tau; P_1$, la acción τ será ejecutada inmediatamente, no permitiendo el paso del tiempo.

2.4 $N[\![P]\!]$ tiene una transición t cuya etiqueta es $\lambda(t) = \tau$, que una vez ejecutada nos lleva al marcado M', que cumple $M'|_{Init([\![P]\!])} = \emptyset$ y $M'|_{S_1} = M_{01}$.
El proceso P a partir del cual se creó $N[\![P]\!]$ ejecutará la acción τ, tras lo cual se comportará como P_1.
Por la hipótesis de inducción tenemos que $(P_1, M_{01}) \in \rho_1$, y por la construcción de ρ tenemos que $(P_1, M') \in \rho$.

- Operador wait: $P = wait(t); P_1$
  Aplicando la hipótesis de inducción al proceso P_1 tenemos que la función ρ_1 establece la equivalencia entre este proceso y su MLTAPN asociada.
  A partir de la función ρ_1 construimos la función ρ que será:

$$\begin{aligned}\rho = \quad & \{(wait(t); P_1, M_0)\} \cup \\ & \{(wait(t - t'); P_1, age(M_0, t')) \mid t' \leq t \} \cup \\ & \{(Q, M) \mid M|_{Init(N[\![P]\!])} = \emptyset, (Q, M|_{S_1}) \in \rho_1\}\end{aligned}$$

  donde S_1 es el conjunto de lugares de la red $N[\![P_1]\!]$.

  Tras definir la relación ρ probemos que se cumplen cada uno de los puntos establecidos por el teorema.

  1. Inmediata.

     2.1 El proceso P permitirá el paso de x unidades de tiempo tal que $x \leq t$. Tras el paso de esta cantidad de tiempo el proceso pasará a comportarse como $wait(t - x); P_1$.

La MLTAPN asociada tendrá una transición etiquetada con τ, cuyo lugar precondición es el lugar inicial de la red. Esta transición no estará habilitada hasta el instante de tiempo t, por lo que se permitirá también el paso de x unidades de tiempo. Una vez transcurrido este tiempo el marcado alcanzado será $age(M_0, x)$. Por la definición de ρ tendremos:

$$(wait(t - t'); P_1, age(M_0, t')) \in \rho$$

Ni P ni $N[\![P]\!]$ permitirán el paso de x unidades de tiempo siendo $x > t$.

2.2 Si el proceso P puede ejecutar una acción será porque es de la forma: $P = wait(0); P_1$; y en ese caso la única acción que podrá ser ejecutada será la acción τ. La MLTAPN tendrá una transición t etiquetada con τ, la cual tendrá un único lugar precondición que será el inicial de la red. El arco (s_0, t) estará etiquetado como $(0, 0)$. El marcado inicial M_0 de la red tendrá solamente un token en el nodo inicial cuya edad será 0, por lo que esa transición estará habilitada para ser ejecutada, y además deberá ser ejecutada en ese instante. Tras su ejecución el marcado M' que se alcanzará cumplirá $M'|_{Init(N[\![P]\!])} = \emptyset$ y $M'|_{S_P \setminus Init(N[\![P]\!])} = M_{01}$.
Por la hipótesis de inducción tenemos que $(P_1, M_{01}) \in \rho_1$ y por tanto:

$$(P_1, M') \in \rho$$

2.3 $N[\![P]\!]$ bajo el marcado M_0 permite el paso de x unidades de tiempo si $x \leq t$, tras lo cual se alcanzará el marcado $age(M_0, x)$. El proceso P, a partir del cual se obtiene $N[\![P]\!]$ también permite el paso de x unidades de tiempo, tras lo cual pasará a comportarse como el proceso $wait(t - x); P_1$.
Por la definición de ρ tendremos:

$$(wait(t - x); P_1, age(M_0, x)) \in \rho$$

Dado que el lugar inicial de $N[\![P]\!]$ posee una transición postcondición etiquetada con τ, la cual estará habilitada en tiempo t, $N[\![P]\!]$ con el marcado M_0 no permitirá el paso de $x \geq t$ unidades de tiempo, y tampoco lo permitirá el proceso P.

2.4 $N[\![P]\!]$ tiene una transición t etiquetada con τ, la cual tendrá un único lugar precondición que será el inicial de la red. Para que esta transición se ejecute con el marcado inicial M_0 el arco (s_0, t) estará etiquetado con $(0, 0)$.

Tras la ejecución de esta transición se alcanzará el marcado M' que cumplirá $M'|_{Init(N[\![P]\!])} = \emptyset$ y $M'|_{S_P \setminus Init(N[\![P]\!])} = M_{01}$.

El proceso P solamente podrá ejecutar una τ por lo que su aspecto será $P = wait(0); P_1$.

Por la hipótesis de inducción tenemos que $(P_1, M_{01}) \in \rho_1$ y por tanto:

$$(P_1, M') \in \rho$$

- Elección Externa: $P = P_1 \Box P_2$

  Para el operador de elección externa hemos proporcionado dos construcciones diferentes, por lo que hemos de distinguir dos casos:

  - Caso 1: $Init(N_1)| = 1 \wedge |Init(N_2)| = 1$

    En este caso debemos comprobar que el proceso P y la red obtenida mediante la aplicación del operador $\Box_1$ son bisimilares.

    Para ellos definimos la función ρ como sigue:

$$\begin{aligned}\rho = \ & \{(P_1 \Box P_2, M_0)\} \cup \\ & \{(P_1' \Box P_2', age(M_0, x)) \mid P_1 \Box P_2 \underset{x}{\longmapsto} P_1' \Box P_2',\ x > 0\} \cup \\ & \{(Q, M) \mid M|_{Init(N)} = \emptyset,\ (Q, M|_{S_1}) \in \rho_1\} \cup \\ & \{(Q, M) \mid M|_{Init(N)} = \emptyset,\ (Q, M|_{S_2}) \in \rho_2\}\end{aligned}$$

    1. Inmediata.

    2.1 Si el proceso P puede dejar pasar x unidades de tiempo es porque tanto P_1 como P_2 pueden dejar pasar esa cantidad de tiempo. Una vez transcurridas las x unidades de tiempo tendremos que el proceso pasará a comportarse como $P = P_1' \Box P_2'$. Aplicando la hipótesis de inducción tenemos que $N[\![P_1]\!]$ y $N[\![P_2]\!]$ con sus respectivos marcados iniciales M_{01} y M_{02} permitirán el paso de x unidades de tiempo, y por tanto $N[\![P]\!]$ también permitirá el paso de esa cantidad de tiempo, tras lo cual el marcado de la red será $age(M_0, x)$.

    Por la definición de ρ tenemos:

$$(P_1' \Box P_2', age(M_0, x)) \in \rho$$

2.2 Si el proceso P ejecuta la acción a es porque la ejecuta el proceso P_1 o el proceso P_2. Consideremos, sin pérdida de generalidad, que la acción la ejecuta el proceso P_1. Así tendremos:

$$P = P_1 \Box P_2 \xrightarrow{a} P_1'$$

Si la acción la ejecutó el proceso P_1, aplicando la hipótesis de inducción tendremos que $(N[\![P_1]\!], M_0)$ puede ejecutar una transición t tal que $\lambda(t) = a$, tras lo cual alcanzará un marcado M_1' que cumple $(P_1', M_1') \in \rho_1$. Además, la relación entre los marcados M_1' y M' es la siguiente:

$$M'|_{Init(N)} = \emptyset \text{ y } M'|_{S_1} = M_1'$$

Por tanto:

$$(P_1', M') \in \rho$$

2.3 Tras el paso de x unidades de tiempo en la red $N[\![P]\!]$ el marcado alcanzado será $age(M_0, x)$.
Si la red $N[\![P]\!]$ con su marcado inicial M_0 puede envejecer es porque tanto la red $N[\![P_1]\!]$ como la red $N[\![P_2]\!]$ con sus respectivos marcados iniciales lo pueden hacer. Entonces, aplicando la hipótesis de inducción tendremos que $P_1 \underset{x}{\longmapsto} P_1'$ y $P_2 \underset{x}{\longmapsto} P_2'$ y por tanto $P_1 \Box P_2 \underset{x}{\longmapsto} P_1' \Box P_2'$.
Por la definición de ρ:

$$(P_1' \Box P_2', age(M_0, x)) \in \rho$$

2.4 Tras ejecutar la transición t, el marcado de la red $N[\![P]\!]$ que alcanzaremos M' cumplirá que $M'|_{Init(N[\![P]\!])} = \emptyset$.
Si la red $N[\![P]\!]$ pueda realizar la transición t es porque esta transición puede ser realizada por $N[\![P_1]\!]$ o por $N[\![P_2]\!]$. Consideremos, sin pérdida de generalidad, que es ejecutada por $N[\![P_1]\!]$, tras lo cual el marcado alcanzado por esa red será M_1', el cual cumple $M'|_{S_1} = M_1'$.
Aplicando la hipótesis de inducción tendremos que $P_1 \xrightarrow{\lambda(t)} P_1'$ y $(P_1', M_1') \in \rho_1$.
Por la definición de la función ρ tenemos:

$$(P_1', M') \in \rho$$

- Caso 2: $|Init(N1)| > 1 \ \vee \ |Init(N_2)| > 1$

  Comprobaremos en este caso que la red obtenida por el operador $\Box_2$ y el proceso P son bisimilares.

  Aplicando la hipótesis de inducción sobre P_1 y P_2 obtendremos respectivamente las relaciones ρ_1 y ρ_2, a partir de las cuales construimos la relación ρ siguiente:

$$\begin{aligned}\rho = \ & \{(P_1 \Box P_2, M_0)\} \cup \ \rho' \ \cup \ \rho'' \\ & \cup \\ & \{(P_1' \Box P_2', age(M_0, x)) \mid P_1 \Box P_2 \underset{x}{\longmapsto} P_1' \Box P_2', \ x > 0\}\end{aligned}$$

  donde

$$\begin{aligned}&\rho' = \{(Q, M) \mid M|_{S_2 \setminus Init(N_2)} = \emptyset, \ (Q, M|_{S_1}) \in \rho_1, \\ &(\exists R_1 \subseteq Init(N_1)^\bullet, R_1 \neq \emptyset, \ \forall t_1 \in R_1, \ \forall t_2 \in Init(N_2)^\bullet, \ M(t_1, t_2) = \emptyset) \\ &\wedge \exists x \geq 0, \ ((\forall s_2 \in Init(N_2), \ M(s_2) = \{x\}) \wedge \\ &\qquad (\forall s \in Init(N[\![P]\!]) \ : \ M(s) = \{x\} \ \vee \ M(s) = \emptyset)) \wedge \\ &\qquad (\forall s_1 \in Init(N_1), \ M|_{S_1}(s_1) \neq \emptyset \Rightarrow \forall t_1 \in s_1^\bullet, \\ &\qquad \forall t_2 \in Init(N_2)^\bullet, \ M(t_1, t_2) = \{x\}))\}\end{aligned}$$

  y

$$\begin{aligned}&\rho'' = \{(Q, M) \mid M|_{S_1 \setminus Init(N_1)} = \emptyset, \ (Q, M|_{S_2}) \in \rho_2, \\ &(\exists R_2 \subseteq Init(N_2)^\bullet, R_2 \neq \emptyset, \ \forall t_2 \in R_2, \ \forall t_1 \in Init(N_1)^\bullet, \ M(t_1, t_2) = \emptyset) \\ &\wedge \exists x \geq 0, \ (\forall s_1 \in Init(N_1), \ M(s_1) = \{x\}) \wedge \\ &\qquad (\forall s \in Init(N[\![P]\!]) \ : \ (M(s) = \{x\} \ \vee \ M(s) = \emptyset)) \wedge \\ &\qquad (\forall s_2 \in Init(N_2), \ M|_{S_2}(s_2) \neq \emptyset \Rightarrow \forall t_2 \in s_2^\bullet, \\ &\qquad \forall t_1 \in Init(N_1)^\bullet, \ M(t_1, t_2) = \{x\}))\}\end{aligned}$$

  donde $N_i = N[\![P_i]\!]$ y S_i es el conjunto de lugares de $N[\![P_i]\!]$. Nótese que una vez que hemos disparado la primera transición t_1 de N_1, todos los lugares (t_1, t_{2j}) se convierten en no marcados, lo cual deshabilita todas las transiciones iniciales de N_2, pero todavía podemos disparar las mismas transiciones de N_1 que podrían ser disparadas considerando la red aislada.

  Una vez definida esta relación comprobaremos que se cumplen los puntos establecidos por el teorema.

  1. Inmediata.

  2.1 El proceso P deja pasar x unidades de tiempo por lo que tanto P_1 como P_2 permiten también el paso de esa cantidad de

tiempo, tras lo cual pasarán a comportarse como P_1' y P_2' respectivamente. Por tanto: $P \underset{x}{\longmapsto} P' = P_1' \Box P_2'$.

Aplicando la hipótesis de inducción sobre P_1 tendremos que $N[\![P_1]\!]$ con el marcado inicial M_{01} permite el paso de x unidades de tiempo, tras lo cual alcanzará el marcado $age(M_{01}, x)$.

Razonando de igual forma sobre P_2 tenemos que ésta alcanzará el marcado $age(M_{02}, x)$.

Si $N[\![P_1]\!]$ y $N[\![P_2]\!]$ con sus respectivos marcados iniciales permiten el paso de las x unidades de tiempo, $N[\![P]\!]$ con su marcado inicial M_0 lo permitirá también alcanzando el marcado $age(M_0, x)$.

Por la definición de ρ podemos concluir:

$$(P_1' \Box P_2', age(M_0, x)) \in \rho$$

2.2 El proceso P realizará la acción a porque la ejecutará el proceso P_1 o el proceso P_2. Consideramos, sin pérdida de generalidad, que la acción es ejecutada por P_1, por lo que tendremos $P = P_1 \Box P_2 \xrightarrow{a} P_1'$.

Aplicando la hipótesis de inducción tenemos que $N[\![P_1]\!]$ con el marcado M_{01} puede ejecutar una transición t etiquetada con a, tras lo cual alcanzará el marcado M_1', y cumplirá $(P_1', M_1') \in \rho_1$.

Si $(N[\![P_1]\!], M_{01})$ permite la ejecución de la transición t tendremos que $N[\![P]\!]$ también la podrá ejecutar alcanzando el marcado M', el cual, por la estructura de la red y la semántica de las redes de Petri, cumplirá $M'|_{S_1} = M_1'$, $\forall\, t_1 \in Init(N_1)^\bullet$, $\forall\, t_2 \in Init(N_2)^\bullet$ $M'(t_1, t_2) = \emptyset$ y $M'|_{S_2} = M_{02}$.

Por tanto podemos concluir:

$$(P', M') \in \rho$$

2.3 La red $N[\![P]\!]$ con el marcado M_0 permite el paso de x unidades de tiempo, tras lo cual alcanzará el marcado $age(M_0, x)$. Si $N[\![P]\!]$ permite el paso de esta cantidad de tiempo, tanto $N[\![P_1]\!]$ como $N[\![P_2]\!]$ con los marcados iniciales M_{01} y M_{02} lo permitirán también alcanzando respectivamente los marcados $age(M_{01}, x)$ y $age(M_{02}, x)$.

Aplicando la hipótesis de inducción sobre ambas redes tendremos se $P_1 \overset{x}{\longmapsto} P_1'$ y $P_2 \overset{x}{\longmapsto} P_2'$, y aplicando la regla $R5c$ de la semántica operacional de $TPAL$ tendremos $P_1 \Box P_2 \overset{x}{\longmapsto} P_1' \Box P_2'$. Por la definición de ρ:

$$(P_1' \Box P_2', age(M_0, x)) \in \rho$$

2.4 $(N[\![P]\!], M_0)$ puede ejecutar una transición t tal que $\lambda(t) = a$, tras lo cual se alcanzará un marcado M'
Si $N[\![P]\!]$ puede ejecutar la transición t es porque $(N[\![P_1]\!], M_{01})$ ó bien $(N[\![P_2]\!], M_{02})$ podrán ejecutarla. Consideraremos, sin pérdida de generalidad, que lo hace $(N[\![P_1]\!], M_{01})$, de modo que tras su ejecución se alcanza el marcado M_1'.
Aplicando la hipótesis de inducción tendremos que $P_1 \xrightarrow{a} P_1'$ de modo que $(P_1', M_1') \in \rho_1$. Por tanto $P = P_1 \Box P_2 \xrightarrow{a} P' = P_1'$.
La relación existente entre los marcados M' y M_1' es la siguiente: $M'|_{S_1} = M_1'$, $\forall\, t_1 \in Init(N_1)^\bullet, \forall\, t_2 \in Init(N_2)^\bullet$ $N'(t_1, t_2) = \emptyset$ y $M'|_{S_2} = M_{02}$. Por tanto:

$$(P', M') \in \rho$$

- Paralelo: $P = P_1 \|_A P_2$
Sean ρ_1 y ρ_2 las relaciones obtenidas para P_1 y P_2 respectivamente mediante la aplicación de la hipótesis de inducción. Para el operador paralelo definimos la relación ρ como sigue:

$$\rho = \{(Q_1 \|_A Q_2, M) \mid (Q_1, M|_{S_1}) \in \rho_1, (Q_2, M|_{S_2}) \in \rho_2\}$$

donde S_i es el conjunto de lugares de $N[\![P_i]\!], i = 1, 2$.

1. Inmediata.

2.1 El paso de x unidades de tiempo en el proceso P se permite porque tanto el proceso P_1 como el proceso P_2 lo permiten, de modo que para el proceso P_1 la evolución será $P_1 \overset{x}{\longmapsto} P_1'$ y para el proceso P_2 tendremos $P_2 \overset{x}{\longmapsto} P_2'$. Aplicando la hipótesis de inducción a ambos procesos tendremos que el marcado alcanzable en $N[\![P_1]\!]$ es $age(M_{01}, x)$ y en $N[\![P_2]\!]$ es $age(M_{02}, x)$, y que además se cumple $(P_1', age(M_{01}, x)) \in \rho_1$ y $(P_2', age(M_{02}, x)) \in \rho_2$.

Si tanto $N[\![P_1]\!]$ como $N[\![P_2]\!]$ con sus marcados iniciales permiten el paso del tiempo, $N[\![P]\!]$ con el marcado M_0 también lo permite, tras lo cual se tiene el marcado M que cumple $M|_{S_1} = age(M_{01}, x)$ y $M|_{S_2} = age(M_{02}, x)$ por lo que podemos concluir:

$$(P_1' \|_A P_2', M) \in \rho$$

2.2 Para comprobar este punto debemos distinguir dos casos dependiendo de la acción ejecutada.

- Caso a ($a \notin A$)
  Si el proceso P ejecuta la acción a es porque la ejecuta uno de los dos procesos P_1 ó P_2. Supongamos, sin pérdida de generalidad, que la ejecuta el proceso P_1, con lo que la evolución del proceso P será $P = P_1 \|_A P_2 \xrightarrow{a} P_1' \|_A P_2$.
  Aplicando la hipótesis de inducción a $N[\![P_1]\!]$ con el marcado M_{01}, obtenemos que puede ejecutarse una transición t etiquetada con $\lambda(t) = a$, de modo que tras su ejecución la red alcanzará el marcado M_1', con:

  $$(P_1', M_1') \in \rho_1$$

  Por la definición del marcado para la red $N[\![P]\!]$, tenemos que el marcado M' alcanzado tras ejecutar esa misma transición en $N[\![P]\!]$ cumple que $M|_{s_1} = M_1'$. Por lo tanto podemos concluir:

  $$(P_1' \|_A P_2, M') \in \rho$$

- Caso b ($a \in A$)
  En este caso la acción es ejecutada por ambos procesos, por lo que tenemos que la evolución del proceso P será:

  $$P_1 \|_a P_2 \xrightarrow{a} P_1' \|_A P_2'$$

  Aplicando la hipótesis de inducción sobre ambos procesos tenemos que $N[\![P_1]\!]$ con el marcado M_{01} ejecutará una transición t tal que $\lambda_1(t) = a$, tras lo cual alcanzará un marcado M_1' tal que $(P_1', M_1') \in \rho_1$ y que $N[\![P_2]\!]$ con M_{02} ejecutará una transición t tal que $\lambda_2(t) = a$, con la que alcanzará un marcado M_2' que cumple $(P_2', M_2') \in \rho_2$.

Por la construcción de $N[\![P]\!]$ tenemos que cualquier marcado, y en particular el alcanzado tras la ejecución de la transición correspondiente a la sincronización cumplirá que $M'|_{s_1} = M'_1$ y $M'|_{s_2} = M'_2$.

Por tanto podemos concluir:

$$(P'_1 \|_A P'_2, M') \in \rho$$

2.3 Tras el paso de x unidades de tiempo, el marcado alcanzado por $N[\![P]\!]$ será $age(M_0, x)$, que cumplirá $age(M_0, x)|_{S_1} = age(M_{01}, x)$ y $age(M_0, x)|_{S_2} = age(M_{02}, x)$.

Como la red $N[\![P]\!]$ permite el paso de x unidades de tiempo, las redes $N[\![P_1]\!]$ y $N[\![P_2]\!]$ también lo permiten. Si aplicamos la hipótesis de inducción sobre la red $N[\![P_1]\!]$ tenemos:

$$P_1 \underset{x}{\longmapsto} P'_1 \text{ tal que } (P'_1, age(M_{01}, x)) \in \rho_1$$

y aplicándola sobre $N[\![P_2]\!]$ tenemos:

$$P_2 \underset{x}{\longmapsto} P'_2 \text{ tal que } (P'_2, age(M_{02}, x)) \in \rho_2$$

Por la definición de ρ podemos concluir:

$$(P'_1 \|_A P'_2, age(M, x)) \in \rho$$

2.4 Para comprobar este punto, al igual que hicimos en el punto 2.2, distinguiremos dos casos dependiendo de la acción ejecutada.

- Caso a ($a \notin A$)

  Para que la red $N[\![P]\!]$ pueda ejecutar una transición t tal que $\lambda(t) = a$ debe ocurrir que, o bien $N[\![P_1]\!]$ bajo el marcado M_{01}, o bien $N[\![P_2]\!]$ bajo el marcado M_{02} puedan ejecutarla. Tras la ejecución de esa transición el marcado alcanzado sera M'.

  Consideremos, sin pérdida de generalidad, que es $N[\![P_1]\!]$ la que ejecuta la transición. Tras la ejecución de la transición la red alcanzará un marcado M'_1. Aplicando la hipótesis de inducción tenemos:

  $$P_1 \xrightarrow{a} P'_1 \text{ tal que } (P'_1, M'_1) \in \rho_1$$

  Para los marcados alcanzables tanto en $N[\![P]\!]$ como en $N[\![P_1]\!]$ y en $N[\![P_2]\!]$ se cumple que $M'|_{S_1} = M'_1$ y $M'|_{S_2} = M'_2$. Por tanto podemos concluir:

  $$(P'_1 \|_A P_2, M') \in \rho$$

- Caso b ($a \in A$)

  Si existe una transición t tal que $\lambda(t) = a$ que es ejecutada por $N[\![P]\!]$ bajo el marcado M_0 es porque en $N[\![P_1]\!]$ existe una transición t_1 tal que $\lambda(t_1) = a$ que puede ser ejecutada bajo el marcado M_{01} y que nos permite alcanzar el marcado M_1'; y en $N[\![P_2]\!]$ también existe una transición t_2 tal que $\lambda(t_2) = a$ que puede ser ejecutada bajo el marcado M_{02}, de modo que se alcanza el marcado M_2'.

  Aplicando la hipótesis de inducción sobre $N[\![P_1]\!]$ tenemos que $P_1 \xrightarrow{a} P_1'$ de modo que $(P_1', M_1') \in \rho_1$. Aplicando de nuevo la hipótesis de inducción, en este caso sobre $N[\![P_2]\!]$, tenemos que $P_2 \xrightarrow{a} P_2'$ y $(P_2', M_2') \in \rho_2$.

  A partir de los marcados M_1' y M_2' podemos obtener fácilmente el marcado M' que será alcanzado en $N[\![P]\!]$ y el cual cumplirá $M'|_{S_1} = M_1'$ y $M'|_{S_2} = M_2'$ por lo que podemos concluir:

$$(P_1' \|_A P_2', M') \in \rho$$

- Ocultamiento $P = P_1 \backslash a$

  Dada la relación ρ_1 obtenida para el proceso P_1 definimos:

$$\rho = \{(Q \backslash a, M) \mid (Q, M) \in \rho_1\}$$

1. Inmediata.

2.1 El proceso P permitirá el paso de x unidades de tiempo, si el proceso P_1 lo permite, de modo que la evolución de P_1 será $P_1 \overset{t}{\longmapsto} P_1'$. Aplicando la hipótesis de inducción sobre el proceso P_1 tenemos que $N[\![P_1]\!]$ con el marcado M_{01} permite también el paso de x unidades de tiempo, tras lo cual alcanzará el marcado $age(M_{01}, x)$, cumpliéndose:

$$(P_1', age(M_{01}, x)) \in \rho_1$$

Por tanto

$$(P_1' \backslash a, age(M_{01}, x)) \in \rho$$

2.2 Distinguiremos dos casos dependiendo de la acción ejecutada.

- El proceso P puede ejecutar una acción $b \neq a$, en tal caso el proceso P_1 puede ejecutar también esa acción. Por tanto la evolución de P_1 será $P_1 \xrightarrow{b} P_1'$. Aplicando la hipótesis de

inducción sobre el proceso P_1 tenemos que $N[\![P_1]\!]$ bajo el marcado M_{01} puede ejecutar una transición t tal que $\lambda(t) = b$. Después de la ejecución de esa transición $N[\![P_1]\!]$ alcanzará un marcado M_1', que cumple $(P_1', M_1') \in \rho_1$. Por tanto:

$$(P_1'\backslash a, M_1') \in \rho$$

- El proceso P ejecuta una acción τ proveniente del ocultamiento de una acción a. Entonces el proceso P_1 puede ejecutar la acción a correspondiente. Por tanto la evolución de P_1 será $P_1 \xrightarrow{a} P_1'$. Aplicando la hipótesis de inducción sobre el proceso P_1 tenemos que $N[\![P_1]\!]$ bajo el marcado M_{01} puede ejecutar una transición t tal que $\lambda(t) = a$. Después de la ejecución de esa transición $N[\![P_1]\!]$ alcanzará el marcado M_1' que cumple $(P_1', M_1') \in \rho_1$. Por tanto:

$$(P_1'\backslash a, M_1') \in \rho$$

2.3 $N[\![P]\!]$ bajo el marcado M_0 permitirá el paso de x unidades de tiempo si $N[\![P_1]\!]$ bajo el marcado M_{01} lo permite. Esta red tras el paso de x unidades de tiempo alcanzará el marcado $age(M_{01}, x)$. Aplicando la hipótesis de inducción tendremos que el proceso P_1 podrá realizar la evolución $P_1 \underset{t}{\longmapsto} P_1'$, de modo que:

$$(P_1', age(M_{01}, x)) \in \rho_1$$

Por tanto:

$$(P_1'\backslash a, age(M_{01}, x)) \in \rho$$

2.4 De nuevo distinguimos dos casos dependiendo de la acción ejecutada.

- Si $N[\![P]\!]$ bajo el marcado M_0 puede ejecutar una transición t tal que $\lambda(t) = b \neq a$ es porque $N[\![P_1]\!]$ bajo el marcado M_{01} también puede ejecutarla. Tras ejecutar la transición el marcado alcanzado por $N[\![P_1]\!]$ será M_1'. Aplicando la hipótesis de inducción tendremos que es posible la evolución $P_1 \xrightarrow{b} P_1'$ y además $(P_1', M_1') \in \rho_1$. Por tanto $(P_1'\backslash a, M_1') \in \rho$
- $N[\![P]\!]$ ejecuta una acción τ proveniente del ocultamiento de una acción a. Entonces $N[\![P_1]\!]$ bajo el marcado M_{01} puede ejecutar una transición t tal que $\lambda(t) = a$, tras lo cual alcanza el

marcado M_1'. Aplicando la hipótesis de inducción sobre el proceso $N[\![P_1]\!]$ tenemos que P_1 puede ejecutar una acción a, tras lo cual pasa a comportarse como P_1, cumpliéndose $(P_1', M_1') \in \rho_1$. Por tanto: $(P_1'\backslash a, M_1') \in \rho$.

□

6.2. Caso Infinito

Tras definir la traducción para procesos finitos, consideremos ahora procesos recursivos de *RTPAL*. El tratamiento de los términos recursivos requiere establecer en principio ciertas restricciones en los términos que intervienen en la elección externa, para que esta construcción funcione correctamente. Al igual que en la definición de los grafos de estados dinámicos, la restricción que debemos establecer es que los lugares iniciales de las redes de Petri asociadas a los procesos argumento de una elección externa no deben tener arcos de entrada.

Para conseguir que esta restricción se cumpla aplicaremos de nuevo la operación *Unfold*, ya definida en el capítulo 4, sobre los procesos que intervienen en la elección externa.

Así, la definición de la MLTAPN para el proceso $P = P_1 \Box P_2$, será la siguiente:

$$N[\![P_1 \Box P_2]\!] = N[\![\mathit{Unfold}(P_1)]\!] \,\Box_i\, N[\![\mathit{Unfold}(P_2)]\!]$$

donde los operadores $\Box_1$ y $\Box_2$ son exactamente los mismos que los definidos para el caso finito (ver Def. 44 y Def. 45).

6.2.1. Recursión

Para construir la MLTAPN asociada a un proceso regular $\mu X.P$ seguiremos los pasos siguientes:

1. Primero construimos la red correspondiente a $P(X)$, considerando para el identificador X un único lugar, tal y como se definió para el proceso *stop*. Nótese que estamos tratando con términos regulares, por lo que no encontraremos términos de la forma $X\backslash a$, $X \Box Q$ ó $X \|_A Q$. Además, esos X-lugares tendrán sólo una transición precondición, ya que cada una de las apariciones de X debe estar prefijada. Denotaremos por S^X al conjunto de X-lugares.

2. Después, eliminamos esos X-lugares, y conectamos todas sus transiciones precondición a todos los lugares iniciales de la red.

Ejemplo 31 Consideremos el proceso:

$$P = a < 0, 0 >; stop \ \Box \ \mu X.(b < 1, 5 >; stop \ \Box \ c < 0, 3 >; X)$$

El segundo argumento de esta elección externa es un término recursivo, por lo que debemos *desplegar* dicha recursión, de modo que el proceso quedará:

$$a < 0, 0 >; stop \ \Box \ (b < 1, 5 >; stop \ \Box \ (c < 0, 3 >; \mu X.(b < 1, 5 >; stop \ \Box \ c < 0, 3 >; X)))$$

Una vez desplegado este término, podemos aplicar la definición anterior, con lo que obtendremos la MLTAPN mostrada en la Figura 6.4.

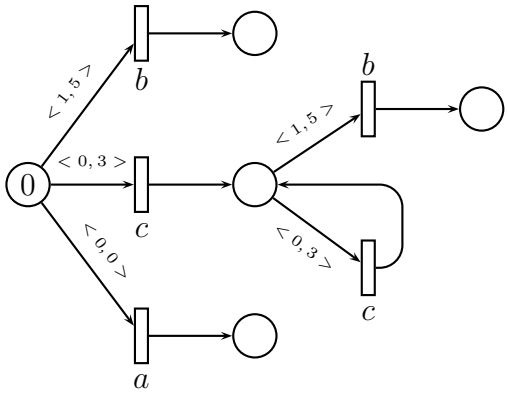


Figura 6.4: Red de Petri para el proceso del ejemplo 31

□

Lema 3 Para todo proceso $P \in RTPAL$, $N[\![P]\!]$ es segura.

Demostración: Esto es una consecuencia directa de que P sea regular.

□

Observemos que si una elección externa aparece en el ámbito de una recursión, no podrá haber conductas paralelas en ninguno de los argumentos,es decir, solamente el operador $\Box_1$ es aplicable en este caso. Esta es una observación importante, ya que la aplicación de $\Box_2$ dentro de una recursión podría ser problemática, ya que $\Box_2$ crea algunos lugares nuevos, y algunos de ellos mantienen tokens después de la resolución del conflicto.

Teorema 5 Sea $P \in RTPAL$ un proceso regular cualquiera y (N, M_0) su correspondiente MLTAPN, obtenida de acuerdo a la construcción anterior.

El sistema de transiciones etiquetado asociado con P, $lts(P)$, y el grafo de alcanzabilidad de (N, M_0) son bisimilares.

Demostración:
Al igual que en el caso finito procederemos por inducción estructural. En la demostración del teorema 4 para el caso finito se realizó el razonamiento para todos los operadores del caso finito, por lo que aquí solamente debemos centrarnos en la construcción de ρ para la recursión.

Sea $P = \mu X.Q_1$, un proceso recursivo y ρ_1 la bisimulación obtenida para $Q_1(X)$ por la hipótesis de inducción (tomando la red asociada a este proceso de acuerdo con el primer paso de la construcción). Entonces tenemos que:

$$\rho = \{(\mu X.Q_1, M_0)\} \cup \{(Q\{\mu X.Q_1/X\}, M|_{S_1 \setminus S_1^X}) \mid (Q, M) \in \rho_1, Q \neq X\}$$

donde S_1 es el conjunto de lugares de $N[\![Q_1(X)]\!]$ y S_1^X es el conjunto de X-lugares en S_1. Nótese que sólo puede existir un X-lugar marcado en $N[\![Q_1(X)]\!]$ en el mismo instante ya que P es regular. Además ningún otro lugar de $N[\![Q_1(X)]\!]$ estará marcado en este caso.

1. Inmediata.

   2.1 El proceso P permite el paso de x unidades de tiempo, de ahí obtenemos que Q_1 también lo permite, tras lo cual pasará a comportarse como Q_1'. Aplicando la hipótesis de inducción tendremos que la red $N[\![Q_1]\!]$ bajo el marcado M_{0,Q_1} permite el paso de x unidades de tiempo, tras lo cual se alcanzará el marcado $age(M_{0,Q_1}, x)$, de modo que:

   $$(Q_1', age(M_{0,Q_1}, x)) \in \rho_1$$

   Por la definición de ρ,

   $$(Q_1'\{\mu X.Q_1'/X\}, age(M_{0,Q_1}, x)|_{S_1 \setminus S_1^X}) \in \rho$$

   2.2 La ejecución de la acción a por parte del proceso P conlleva que Q_1 también puede ejecutarla, tras lo cual pasa a comportarse como Q_1'.

Aplicando la hipótesis de inducción tenemos que $N[\![Q_1]\!]$ con el marcado M_{0,Q_1} permite la ejecución de una transición t tal que $\lambda(t) = a$, alcanzando el marcado M'_{Q_1}.

Si $Q'_1 = X$ los lugares marcados en $N[\![Q_1]\!]$ serán X-lugares, por lo que en $N[\![P]\!]$ el marcado alcanzado será M_0, que será igual a M_{0,Q_1}, y por tanto:

$$(\mu X.Q'_1, M_{0,Q_1}) \in \rho$$

Si $Q'_1 \neq X$ los lugares marcados pertenecerán también a $N[\![P]\!]$, y por tanto:

$$(Q'_1, M'_{Q_1}) \in \rho$$

2.3 La red $N[\![P]\!]$ con el marcado M_0 permite el paso de x unidades de tiempo, tras lo cual se alcanza el marcado $age(M_0, x)$. Por tanto $N[\![Q_1]\!]$ con el marcado M_{0,Q_1} también permitirá el paso de esa cantidad de tiempo alcanzando el marcado $age(M_{0,Q_1}, x)$, cumpliéndose además $age(M_{0,Q_1}, x) = age(M_0, x)|_{S_1 \setminus S_1^X}$.

Aplicando la hipótesis de inducción sobre $N[\![Q_1]\!]$ tendremos que $Q_1 \underset{x}{\longmapsto} Q'_1$ y $(Q'_1, age(M_{0,Q_1}, x)) \in \rho_1$.

Por tanto podemos concluir:

$$(Q'_1\{\mu X.Q'_1/X\}, age(M_0, x)) \in \rho$$

2.4 La red $N[\![P]\!]$ con el marcado M_0 puede ejecutar una transición t, de modo que se obtendrá el marcado M'. Entonces $N[\![Q_1]\!]$ con el marcado M_{0,Q_1} también podrá ejecutar esa transición alcanzando el marcado M'_{Q_1}.

Los marcados obtenidos en ambas redes estarán relacionados de modo que si los lugares marcados en M'_{0,Q_1} no son X-lugares tendremos que $M' = M'_{Q_1}|_{S_1 \setminus S_1^X}$.

Si los lugares marcados son X-lugares, en realidad solamente existirá un lugar marcado, y entonces en $N[\![P]\!]$ se marcarán todos los lugares iniciales, es decir $M' = M_0$.

Por tanto podemos concluir que:

$$(Q'_1\{\mu X.Q'_1/X\}, M'_{Q_1}) \in \rho$$

□

6.3. Más allá de los términos regulares

La construcción presentada anteriormente establece que los términos sobre los que puede ser aplicada deben ser regulares, lo que impone ciertas restricciones, que nos gustaría poder soslayar, al menos en ciertos casos particulares. De hecho, alguna de las restricciones impuestas puede ser eliminada con relativa sencillez.

- Una primera restricción que puede ser eliminada es la prohibición de la existencia del operador de ocultamiento dentro de una recursión. En este caso, cuando generemos la red para $P(X)$ obtendremos un conjunto de (X, A)-lugares, los cuales son esencialmente X-lugares, donde todas las acciones del conjunto A están ocultadas para X. Por tanto, podemos *replicar* $N[\![P(X)]\!]$ para cada (X, A)-lugar, ocultando en esas redes las acciones del conjunto A, y entonces enlazar las precondiciones de cada (X, A)-lugar con los lugares iniciales de la red correspondiente.

  Ejemplo 32 Consideremos el proceso:

  $$\mu X.(a < 1, 3 >; ((c < 1, 4 >; X)\Box((a < 0, 0 >; X)\backslash a)))$$

  La MLTAPN correspondiente es la mostrada en la Figura 6.5.

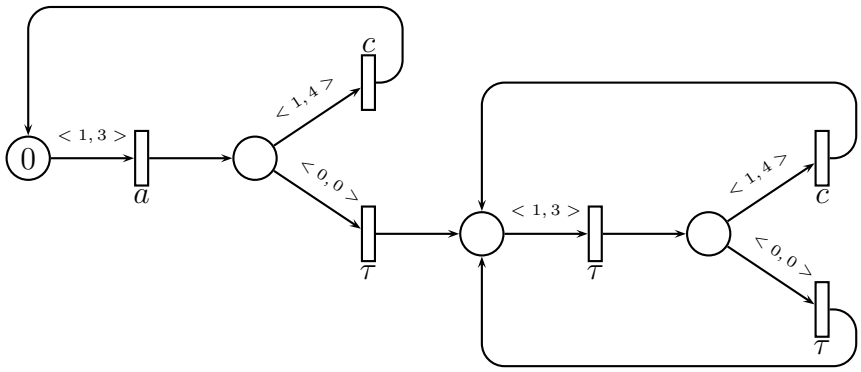


Figura 6.5: MLTAPN correspondiente al proceso del ejemplo 32

□

- La restricción de no permitir paralelismo dentro de la recursión no puede ser eliminada en general, ya que se podrían generar redes infinitas. Sin embargo,

en algunos casos particulares sí que podemos levantar esta restricción y aplicar la construcción anterior para ciertos procesos recursivos con subtérminos paralelos. En estos casos obtendremos redes que no serán seguras, pero sí que serán débilmente seguras.

Ejemplo 33 Consideremos, por ejemplo, el proceso:

$$P = \mu X.(a < 5, 8 >; X \parallel_{\emptyset} b < 1, 2 >; c < 0, 1 >; stop)$$

Cuando la componente izquierda cicla, tendremos que la parte derecha o bien ha terminado o está muerta. En consecuencia, la red que podemos asociar a este proceso (ver Figura 6.6) no será segura, pero sí que será débilmente segura, ya que existirán algunos tokens muertos en la subred correspondiente a la componente derecha cuando se coloquen nuevos tokens en los lugares iniciales.

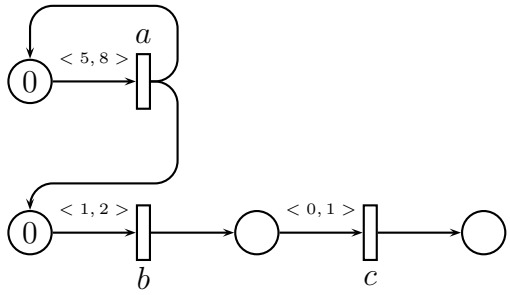


Figura 6.6: MLTAPN del proceso del ejemplo 33

□

Como podemos comprobar en los ejemplos anteriores, podemos extender la aplicación de la construcción definida para términos recursivos a algunos términos que no sean regulares. Esta construcción será aplicable en particular a aquellos términos recursivos $\mu X.P$ que cumplen las siguientes restricciones:

1. No existen subtérminos recursivos en P.

2. Cada ocurrencia de X en P está prefijada.

3. El ocultamiento no está permitido en P. Volvemos a considerar de nuevo esta restricción por simplicidad, pero podría ser eliminada mediante replicaciones, de igual forma a lo indicado al inicio de esta subsección.

4. Para cada subtérmino $P_1 \|_A P_2$ en P en los que X sólo aparece en uno de los componente P_1 ó P_2 se cumple que:

$$Max\_time(P_2) < Min\_time(P_1, X) + Min\_time(P, P_1 \|_A P_2)$$

donde *Max_time* y *Min_time* se definen como sigue:

$$\begin{aligned}
&Max_time(stop) = 0\\
&Max_time(a < x_1, x_2 >; P_1) = x_2 + Max_time(P_1)\\
&Max_time(\tau; P_1) = Max_time(P_1)\\
&Max_time(wait(x); P_1) = x + Max_time(P_1)\\
&Max_time(P_1 \Box P_2) = \text{máx}(Max_time(P_1), Max_time(P_2))\\
&Max_time(P_1 \|_A P_2) = \text{máx}(Max_time(P_1), Max_time(P_2))\\
&Max_time(P \backslash a) = Max_time(P)
\end{aligned}$$

$Max\_time(P)$ es una cota superior para el tiempo que P puede tardar en finalizar su ejecución.

$$\begin{aligned}
&Min_time(Q, Q) = 0\\
&Min_time(stop, Q) = \infty\\
&Min_time(X, Q) = \infty\\
&Min_time(a < x_1, x_2 >; P_1, Q) = x_1 + Min_time(P_1, Q)\\
&Min_time(\tau; P_1, Q) = Min_time(P_1, Q)\\
&Min_time(wait(x); P_1, Q) = x + Min_time(P_1, Q)\\
&Min_time(P_1 \Box P_2, Q) = \text{mín}(Min_time(P_1, Q), Min_time(P_2, Q))\\
&Min_time(P_1 \|_A P_2, Q) = \text{mín}(Min_time(P_1, Q), Min_time(P_2, Q))\\
&Min_time(P \backslash a, Q) = Min_time(P, Q)
\end{aligned}$$

donde, en todos los casos, excepto en el primero, se considera que en la parte izquierda de la definición, el proceso Q es distinto del proceso que aparece como primer argumento.

$Min\_time(P, P')$ es una cota inferior para el tiempo que P tardará en llegar a su subtérmino P' (ó ∞ si no llega).

5. Cada elección externa $P_1 \Box P_2$ del tipo 2 en P satisface:

$$MW(P_1 \Box P_2) < Min\_time(P_1 \Box P_2, X) + Min\_time(P, P_1 \Box P_2)$$

donde *Min_time* se define como en el apartado anterior.

Ejemplo 34 Consideremos el proceso:

$$P = \mu X.(P_1 \|_\emptyset P_2)$$

donde

$$P_1 = a < 5, 8 >; X \text{ y } P_2 = b < 1, 2 >; c < 0, 1 >; stop)$$

Si calculamos *Min_time* y *Max_time* para ese proceso obtenemos: $Max\_time(P_2) = 3$, $Min\_time(P_1, X) = 5$, y $Min\_time(P, P_1 \|_\emptyset P_2) = 0$.

Como se cumple que $Max\_time(P_2) < Min\_time(P_1, X) + Min\_time(P, P_1 \|_\emptyset P_2)$ la construcción será aplicable a este proceso.

□

Lema 4 Sea $P \in fTPAL$ con $Max\_time(P) < \infty$.

Todo marcado alcanzable de $N[\![P]\!]$ en un instante mayor que $Max\_time(P)$ será muerto.

Demostración:
Por inducción estructural podemos comprobar fácilmente que $Max\_time(P)$ es una cota superior para el tiempo que tarda el proceso P en convertirse en el proceso *stop*, es decir, para que el proceso P llegue a bloquearse. Entonces bastará con aplicar el teorema 4.

□

Lema 5 Para todo proceso $P \in TPAL$ que cumple las condiciones 1-5, $N[\![P]\!]$ es débilmente segura.
Demostración:

Podemos reducir la demostración a aquellos procesos recursivos que cumplen las condiciones 1-5. Puede comprobarse por inducción estructural que $Min\_time(P, Q)$ es realmente una cota inferior para el tiempo requerido por P para alcanzar el subtérmino Q. Por definición, los únicos operadores de red, para los cuales puede permanecer algún token no usado en la subred correspondiente cuando se alcanza un X-lugar son $\square_2$ y $\|_A$.

- Con $\square_2$ hemos creado algunos lugares nuevos (t_1, t_2), y algunos de ellos permanecen marcados una vez que la elección ha sido resuelta. Sin embargo, los

arcos que salen de esos lugares están etiquetados por $< 0, MW(P_1 \Box P_2) >$, así que esos tokens serán muertos y no se podrá disparar ninguna transición hasta que una nueva colección de tokens pueda llegar a esos lugares, debido a la condición 5. Los tokens que permanecen en esos lugares iniciales de la subred correspondiente a $P_1 \Box P_2$ también serán muertos, ya que sus arcos deben estar etiquetados con valores $< x_1, x_2 >$, con $x_2 \leq MW(P_1 \Box P_2)$.

- Sea $P_1 \|_A P_2$ un subtérmino de P, y consideremos que el identificador X aparece en P_1.

  De la condición 4 y el lema 4 tenemos que todos los tokens que permanecen en los lugares de la subred $N[\![P_2]\!]$ estarán muertos. Por tanto en esos lugares no aparecerá ningún token vivo hasta que una colección de nuevos tokens llegue a los lugares iniciales de la subred $N[\![P_1 \|_A P_2]\!]$.

  En P_1 puede aparecer un número finito de operadores paralelo, por ello procederemos por inducción sobre este número:

  - Caso base. En P_1 no aparece ningún operador paralelo.

    En este caso los únicos tokens que permanecen en $N[\![P_1]\!]$ después de un ciclo son aquellos correspondientes a una elección externa de tipo 2, para los cuales ya hemos probado que es débilmente segura.

  - Caso general. En P_1 aparecen n operadores paralelos.

    En este caso tendremos un subtérmino $P_{11} \|_B P_{12}$ en P_1, donde en P_{11} aparecen n_1 operadores paralelos y en P_{12} aparecen n_2, siendo $n - 1 = n_1 + n_2$.

    De igual forma podemos descomponer el subtérmino P_{11} sucesivamente hasta llegar a un subtérmino $P_{1n} \|_C P_{2n}$ donde ni P_{1n} ni P_{2n} tienen ningún paralelo, pero que X aparece en alguno de los dos subtérminos. Consideraremos que aparece en P_{1n}.

    De acuerdo con la condición 4 y el lema 4 tenemos que todos los tokens que aparecen en los lugares de P_{2n} estarán muertos, por lo que la subred $N[\![P_{1n} \|_C P_{2n}]\!]$ será débilmente segura.

    Por tanto todas las redes que contienen la subred $N[\![P_{1n} \|_C P_{2n}]\!]$ serán débilmente seguras y $N[\![P]\!]$ será débilmente segura.

□

Corolario 1 Sea $P \in TPAL$ un proceso cualquiera para el cual todos sus subtérminos $\mu X.Q$ cumplen las condiciones 1-5.

El grafo de alcanzabilidad de $N[\![P]\!]$ y el sistema de transiciones etiquetado asociado a P son bisimilares.

Demostración: La relación ρ definida para el caso de los términos regulares nos sirve, aunque en este caso hemos de generalizarla considerando marcados débilmente seguros.

□

Ejemplo 35 Consideremos de nuevo el sistema automático de control de un paso a nivel descrito en el ejemplo 3 del capítulo 2.

De acuerdo con la descripción del sistema realizada allí, y con la especificación del sistema mostrada en el capítulo 3, la red de Petri obtenida para este sistema será la mostrada en la Figura 6.7

□

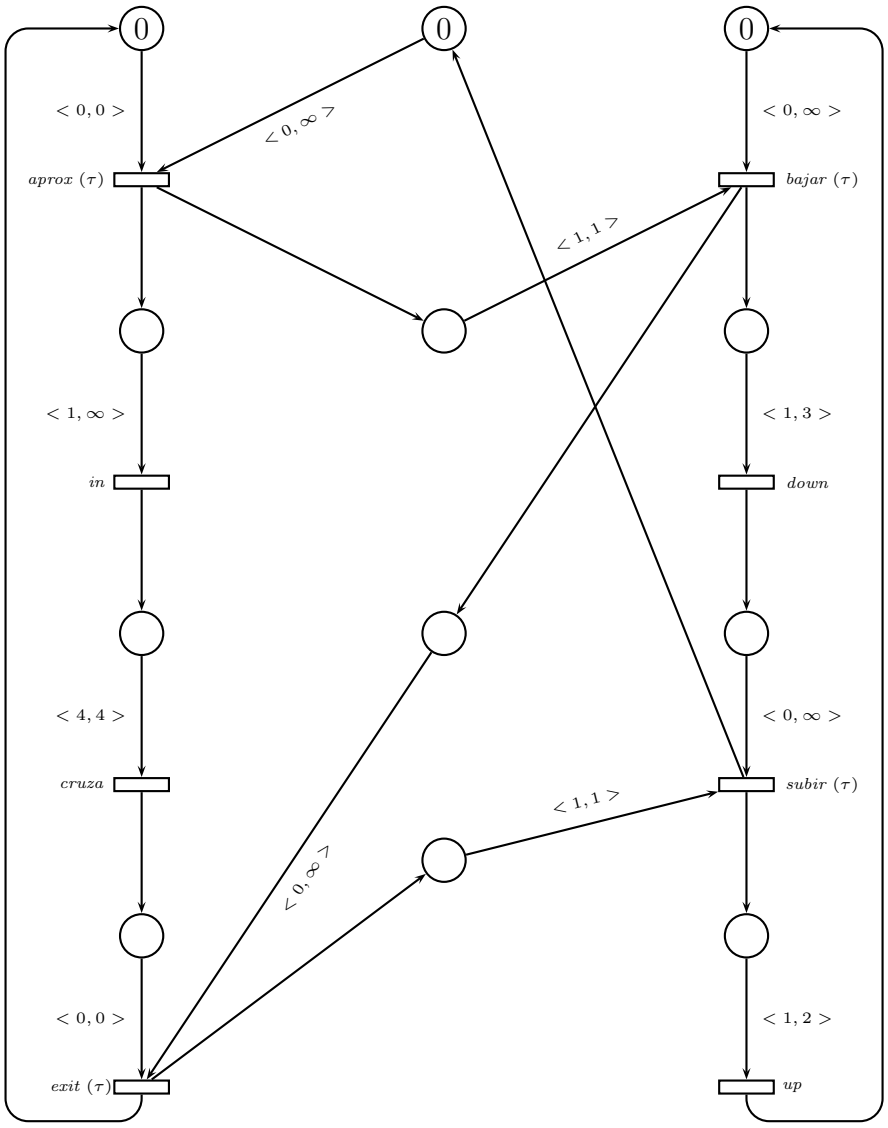


Figura 6.7: MLTAPN obtenida para el sistema de control de un paso a nivel

Capítulo 7

TPPAL. Extensión del álgebra TPAL con probabilidades

En este capítulo presentamos la extensión probabilística de TPAL, a la que hemos llamado TPPAL. Mediante este lenguaje podremos especificar tanto sistemas con comportamientos probabilísticos como sistemas tolerantes a fallos, protocolos de comunicación, sistemas de eventos probabilísticos, etc.

7.1. Sintaxis

La sintaxis de los términos probabilísticos de TPPAL vendrá definida por la siguiente expresión BNF:

$$P ::= \sum_{i \in I} [q_i].N \mid P \parallel_A N \mid N \parallel_A P \mid P \parallel_A P \mid P \backslash a \mid X \mid \mu X.P$$

donde Vis es un conjunto finito de acciones visibles, $Act = Vis \cup \{\tau\}$ y $A \subseteq Vis$. I es un conjunto finito de índices y $q_i \in (0, 1]$. Con la letra N se identifica a cualquier proceso no-determinista definido de acuerdo con la sintaxis de *TPAL* presentada en el capítulo 3.

A continuación procedemos a describir de manera informal el comportamiento de los operadores definidos anteriormente.

$\sum_{i \in I} [q_i].N_i$.- Representa una elección probabilística, donde el sistema decide qué proceso será ejecutado atendiendo a las probabilidades q_i asociadas a cada proceso

N_i.

$N \parallel_A P$, $P \parallel_A N$, $P \parallel_A P$.- Representa la ejecución en paralelo de los procesos argumento, sincronizando sobre las acciones del conjunto A. Los procesos pueden ser no-deterministas o probabilísticos.

$P\backslash a$.- Es el operador de ocultamiento de una acción (transformará la acción a en una acción urgente e interna).

$\mu X.P$.- Es el clásico operador de recursión, para la definición de procesos infinitos.

Una vez definidas por separado la sintaxis de los procesos temporizados y la de los procesos probabilísticos, podemos unir ambas para obtener la sintaxis completa del álgebra TPPAL (Timed and Probabilistic Process ALgebra), la cual vendrá definida por la siguiente expresión BNF:

$$\begin{aligned} T ::= &\quad P \mid N \\ N ::= &\quad stop \mid a < t_1, t_2 >; T \mid \tau; T \mid wait(t); T \mid N \,\Box\, N \mid N \parallel_A N \mid N\backslash\, a \mid X \mid \mu X.N \\ P ::= &\quad \sum_{i \in I} [q_i].N \mid P \parallel_A N \mid N \parallel_A P \mid P \parallel_A P \mid P\backslash a \mid X \mid \mu X.P \end{aligned}$$

NOTACIÓN: Mediante las mayúsculas P y P_i denotaremos los términos probabilísticos, mientras que N y N_i denotarán términos no-deterministas. Con T y T_i se denotará cualquier término del lenguaje.

El conjunto de términos obtenidos mediante la anterior sintaxis lo denotaremos por *TPPAL*, si bien por regla general nos restringiremos a los términos regulares definidos como en el capítulo 4 (son términos cerrados y guardados, en los cuales no se permiten los operadores paralelo y ocultamiento dentro de una recursión). Al conjunto de términos regulares obtenidos mediante esta sintaxis los denotaremos como *RTPPAL*.

7.2. Semántica operacional

La introducción de los términos probabilísticos en el lenguaje hace necesaria la introducción de un nuevo tipo de transición (transición probabilística) para definir la semántica operacional del lenguaje, y por tanto un conjunto de reglas que definan el comportamiento y evolución de los procesos probabilísticos.

Definición 46 (Transición probabilística)
Estas transiciones representan las decisiones que el sistema tomará internamente, de

acuerdo con las probabilidades asignadas a cada uno de los subtérminos. La forma de representar estas transiciones será la siguiente:

$$P \rightarrow_q N \text{ donde } q \in (0,1]$$

Indica que el proceso probabilístico P pasa a comportarse como el proceso no-determinista N con probabilidad q. □

En la Tabla 7.1 podemos ver las nuevas reglas introducidas, en las cuales todos los términos que aparecen son regulares.

P1) $\dfrac{}{\sum_{i \in I} [q_i] N_i \longrightarrow_{q_j} N_j} \quad \forall j \in I$

P2a) $\dfrac{P_1 \longrightarrow_q N_1}{P_1 \parallel_A N_2 \longrightarrow_q N_1 \parallel_A N_2}$ **P2b)** $\dfrac{P_2 \longrightarrow_q N_2}{N_1 \parallel_A P_2 \longrightarrow_q N_1 \parallel_A N_2}$

P2c) $\dfrac{P_1 \longrightarrow_{q_1} N_1 \quad P_2 \longrightarrow_{q_2} N_2}{P_1 \parallel_A P_2 \longrightarrow_{q_1 \times q_2} N_1 \parallel_A N_2}$

P3) $\dfrac{P \longrightarrow_q N}{P \backslash a \longrightarrow_q N \backslash a}$

P4) $\dfrac{P\{\mu x.P/x\} \longrightarrow_q N}{\mu x.P \longrightarrow_q N}$

Tabla 7.1: Reglas Probabilísticas

Obsérvese que con este nuevo marco semántico se establece que las decisiones probabilísticas serán las primeras en resolverse.

Veamos brevemente la interpretación de las nuevas reglas.

La regla P1 captura la semántica clásica del operador de elección probabilística.

El conjunto de reglas $P2$ captura el comportamiento del operador paralelo, y más concretamente las reglas $P2a$ y $P2b$ definen la evolución independiente de cada uno de los procesos mediante una elección probabilística, mientras que la regla $P2c$ establece la evolución conjunta de ambos procesos.

La regla $P3$ describe el comportamiento de un proceso probabilístico en el que se oculta una acción. Este proceso mantiene el mismo comportamiento probabilístico que el proceso de partida. Finalmente, $P4$ es la regla clásica de la recursión.

Las reglas correspondientes a los procesos no-deterministas son las mostradas en la Tabla 7.2, las cuales son esencialmente las mismas que las de la tabla 3.1.

Definición 47 (Semántica Operacional de TPPAL)
Para cada proceso P regular de *TPPAL*, definimos la semántica operacional de P como el multiconjunto de transiciones que pueden ser inferidas aplicando el sistema de reglas definido en las Tablas 7.1 y 7.2, donde cada transición aparece tantas veces como formas diferentes haya de derivarla.

□

Definición 48 (Computación probabilístico-temporizada)
Sean $T,\ T',\ T'' \in TPPAL$, procesos regulares de *TPPAL*, $w \in (Act \cup \mathbb{R}_0^+ \cup \{[q] \mid q \in (0,1]\})^*$ una secuencia temporizada y probabilística (cadena de acciones, tiempos y probabilidades), $e \in Act$, $t \in \mathbb{R}_0^+$ y $q \in (0,1]$.

1. Sea $T \xrightarrow{e} T'$ una posible evolución del proceso T. Entonces $T \stackrel{<e>}{\Longrightarrow}_{0,1} T'$ es una computación probabilístico-temporizada.

2. Sea $T \underset{t}{\longmapsto} T'$ una posible evolución del proceso T. Entonces $T \stackrel{<t>}{\Longrightarrow}_{t,1} T'$ es una computación probabilístico-temporizada.

3. Sea $T \longrightarrow_q N'$ una posible evolución del proceso T. Entonces $T \stackrel{<[q]>}{\Longrightarrow}_{0,q} N'$ es una computación probabilístico-temporizada.

4. Sea $T \stackrel{<w>}{\Longrightarrow}_{t,p} T'$ una computación probabilístico-temporizada, y $T' \xrightarrow{e} T''$ una posible evolución del proceso T'. Entonces $T \stackrel{<w.e>}{\Longrightarrow}_{t,p} T''$ es una computación probabilístico-temporizada.

5. Sea $T \stackrel{<w>}{\Longrightarrow}_{t,p} T'$ una computación probabilístico-temporizada, y $T' \underset{t'}{\longmapsto} T''$ una posible evolución de T'. Entonces $T \stackrel{<w.t'>}{\Longrightarrow}_{t+t',p} T''$ es una computación probabilístico-temporizada.

6. Sea $T \stackrel{<w>}{\Longrightarrow}_{t,p} T'$ una computación probabilístico-temporizada, y $T' \longrightarrow_q T''$ una posible evolución de T'. Entonces $T \stackrel{<w.[q]>}{\Longrightarrow}_{t,p\times q} T''$ es una computación probabilístico-temporizada.

□

Ejemplo 36 Para finalizar este capítulo, especificaremos el protocolo AUY con TPPAL. Éste es un protocolo que asegura una comunicación fiable en un sistema con un canal que puede fallar.

Suponemos que el sistema está compuesto por un emisor y un receptor que se comunican mediante mensajes a través de dos canales no fiables que pueden perder mensajes con una probabilidad $p = 0{,}2$. El retraso de transmisión de mensajes para ambos canales es de 2 unidades de tiempo.

El emisor utiliza el primer canal para enviar los mensajes al receptor. Tras enviar el mensaje queda a la espera de recibir un mensaje del receptor. Si tras enviar el mensaje el primer canal falla, éste detecta la situación de fallo después de 4 unidades de tiempo y reemplaza el mensaje por un mensaje especial λ que envía al receptor.

Una vez que el receptor recibe el mensaje si éste no es el mensaje λ el receptor envía un mensaje *ack* de reconocimiento al emisor a través del segundo canal, el cual puede perderlo con una probabilidad $p = 0{,}2$. En caso de que el mensaje *ack* se pierda en el canal, éste lo reemplaza por un mensaje λ' el cual es enviado al emisor después de 4 unidades de tiempo.

Por otro lado, si el receptor recibe un mensaje λ, enviará un mensaje erróneo λ'' a través del segundo canal, que hará que el canal genere un mensaje λ' después de 3 unidades de tiempo. Al recibir este mensaje, el emisor reenviará el mensaje original a través del primer canal.

Emisor

$T = \quad in < 0, \infty >; T_1$

$T_1 = \quad emsg < 2, 2 >; (rack < 0, \infty >; T \,\square\, \lambda' < 0, \infty >; T_1)$

Canal

$C_1 = \quad emsg < 0, \infty >; ([0, 2]\, \lambda < 4, 4 >; C_1 + [0, 8]\, rmsg < 0, 0 >; C_1)$

$C_2 = \quad (eack < 0, \infty >; ([0, 2]\, \lambda' < 4, 4 >; C_2 + [0{,}8]\, rack < 0, 0 >; C_2)) \,\square$
$\quad (\lambda'' < 0, \infty >; \lambda' < 3, 3 >; C_2)$

Receptor

$R = \quad (rmsg < 0, \infty >; out < 0, \infty >; eack < 2, 2 >; R) \square$
$\quad (\lambda < 0, \infty >; \lambda'' < 2, 2 >; R)$

Protocolo AUY

$AUY = \quad (T \parallel_{\{emsg, rack, \lambda'\}} (C_1 \parallel_{\{\}} C_2)) \parallel_{\{rmsg, eack, \lambda, \lambda''\}} R$

$\square$

N1) $\text{stop} \underset{t}{\longmapsto} \text{stop} \quad \forall t \in \mathbb{R}_0^+$

N2a) $a < 0, t >; T \xrightarrow{a} T \quad \forall t \in \mathbb{R}_0^+$

N2b) $a < t_1, t_2 >; T \underset{t'}{\longmapsto} a < t_1 \dot{-} t', t_2 - t' >; T \;\; 0 \leq t' \leq t_2,$

donde $x \dot{-} y = \max\{0, x - y\}$

N2c) $a < 0, 0 >; T \underset{t'}{\longmapsto} \text{stop} \;\; \forall t' > 0$

N3) $\tau; T \xrightarrow{\tau} T$

N4a) $\text{wait}(t); T \underset{t'}{\longmapsto} \text{wait}(t - t'); T \,, \; 0 \leq t' \leq t$

N4b) $\text{wait}(0); T \xrightarrow{\tau} T$

N5a) $\dfrac{N_1 \xrightarrow{e} T_1}{N_1 \Box N_2 \xrightarrow{e} T_1}$ **N5b)** $\dfrac{N_2 \xrightarrow{e} T_2}{N_1 \Box N_2 \xrightarrow{e} T_2}$

N5c) $\dfrac{N_1 \underset{t'}{\longmapsto} N_1' \quad N_2 \underset{t'}{\longmapsto} N_2'}{N_1 \Box N_2 \underset{t'}{\longmapsto} N_1' \Box N_2'}$

N6a) $\dfrac{N_1 \xrightarrow{e} T_1 \quad e \notin A}{N_1 \parallel_A N_2 \xrightarrow{e} T_1 \parallel_A N_2}$ **N6b)** $\dfrac{N_2 \xrightarrow{e} T_2 \quad e \notin A}{N_1 \parallel_A N_2 \xrightarrow{e} N_1 \parallel_A T_2}$

N6c) $\dfrac{N_1 \xrightarrow{a} T_1, \quad N_2 \xrightarrow{a} T_2 \quad a \in A}{N_1 \parallel_A N_2 \xrightarrow{a} T_1 \parallel_A T_2}$ **N6d)** $\dfrac{N_1 \underset{t'}{\longmapsto} N_1', \quad N_2 \underset{t'}{\longmapsto} N_2'}{N_1 \parallel_A N_2 \underset{t'}{\longmapsto} N_1' \parallel_A N_2'}$

N7a) $\dfrac{N \xrightarrow{e} T' \quad e \in Act \quad e \neq a}{N \backslash a \xrightarrow{e} T' \backslash a}$ **N7b)** $\dfrac{N \xrightarrow{a} T}{N \backslash a \xrightarrow{\tau} T \backslash a}$

N7c) $\dfrac{N \underset{t}{\longmapsto} N', \neg N \xrightarrow{a} \wedge \neg[\exists\, t' < t, \; N \underset{t'}{\longmapsto} N'' \,, \; N'' \xrightarrow{a}]}{N \backslash a \underset{t}{\longmapsto} N' \backslash a}$

N8a) $\dfrac{N\{\mu X.N/X\} \xrightarrow{e} T'}{\mu X.N \xrightarrow{e} T'}$ **N8b)** $\dfrac{N\{\mu X.N/X\} \underset{t}{\longmapsto} N'}{\mu X.N \underset{t}{\longmapsto} N'}$

N9) $\dfrac{N_1 \underset{t_1}{\longmapsto} N_2 \quad N_2 \underset{t_2}{\longmapsto} N_3}{N_1 \underset{t_1+t_2}{\longmapsto} N_3}$

Tabla 7.2: Reglas No-Deterministas

Capítulo 8

Grafos de estados dinámicos probabilísticos

Tras la ampliación del álgebra de procesos con operadores probabilísticos, en este capítulo presentamos la definición de los *grafos de estados dinámicos probabilísticos*, como una extensión de los grafos de estados dinámicos presentados en el capítulo 4, así como la traducción de los términos regulares del álgebra TPPAL a este nuevo tipo de grafos.

Para ello, entre las alternativas posibles, se ha optado por introducir en el grafo un nuevo tipo de nodos que denominaremos *probabilísticos*, y un nuevo tipo de arcos que denominaremos *arcos probabilísticos*. De este modo, en el grafo tendremos por un lado los nodos temporizados, los cuales permitirán el paso del tiempo y la ejecución de transiciones de acción, por lo que de ellos saldrán arcos etiquetados esencialmente con acciones, y los nodos probabilísticos, a partir de los cuales solamente se podrá evolucionar mediante una elección probabilística, por lo que de ellos solamente saldrán arcos etiquetados con probabilidades.

Definición 49 (Grafo de estados dinámico probabilístico)

Un *grafo de estados dinámico probabilístico* es una tupla

$$(NT, NP, ET, EP, \lambda, \gamma, n_0, clocks)$$

donde:

NT.- Es el conjunto de nodos temporizados.

NP.- Conjunto de nodos probabilísticos con $NT \cap NP = \emptyset$.

ET.- Conjunto de arcos temporizados, los cuales están definidos como:

$$ET \subseteq NT \times (NT \cup NP)$$

EP.- Conjunto de arcos probabilísticos, definidos como:

$$EP \subseteq NP \times (NT \cup NP)$$

λ.- Función de etiquetado para los arcos temporizados.

$$\lambda : E \longrightarrow Act_\tau \times Times \times \mathcal{P}(clocks)$$

donde: $Times : clocks \hookrightarrow \mathbb{R}_0^+ \times \mathbb{R}_0^+ \cup \{\infty\}$ es una función parcial que define las restricciones sobre los relojes, de modo que

$$Times(R_i) = (\alpha_i, \beta_i)$$

$\gamma : EP \longrightarrow (0, 1]$.- Función de etiquetado para los arcos probabilísticos. Esta función asignará a cada arco la probabilidad que tiene la transición correspondiente a dicho arco de ser ejecutada. Deberá cumplirse que la suma de las probabilidades asignadas a todos los arcos salientes de un nodo sea 1, es decir:

$$\forall\, n \in NP : \sum_{\substack{e \in EP \\ n \xrightarrow{e}}} \gamma(e) = 1$$

$n_0 \in NT \cup NP$.- Es el nodo inicial del grafo.

clocks.- Conjunto de relojes asociado al grafo.

□

En la representación gráfica de los grafos de estados dinámicos probabilísticos distinguiremos los nodos probabilísticos marcándolos con una cruz en su interior. Asimismo, los arcos probabilísticos se reconocerán por las probabilidades que llevan asociadas, en lugar de las ternas (a, r, S) de los arcos temporizados.

8.1. Semántica

Para formalizar la semántica de los grafos de estados dinámicos probabilísticos tenemos que generalizar en primer lugar la noción de estado.

Definición 50 (Estado)
Dado un grafo de estados dinámico probabilístico $G = (NT, NP, ET, EP, \lambda, \gamma, n_0, clocks)$, definimos un estado del mismo como una tupla (n, C, t) donde

- $n \in NT \cup NP$ es un nodo.
- $C : clocks \longrightarrow \mathbb{R}_0^+$ es la función que indica el valor de cada uno de los relojes en ese estado.
- t es el instante actual. Este valor siempre cumplirá $t \geq C(R_g)$.

□

El estado inicial de los grafos será $s_0 = (n_0, C_0, 0)$, donde $C_0(R_j) = 0$, $\forall\, R_j \in clocks$.

Definición 51 (Estado Probabilísticamente Estable)
Sea $G = (NT, NP, ET, EP, \lambda, \gamma, n_0, clocks)$ un grafo de estados dinámico probabilístico y $s = (n, C, t)$ un estado del mismo.

Se dice que s es un *estado probabilísticamente estable*, lo que denotaremos como $s \downarrow$, si y sólo si $n \in NT$.

Si s no es un estado probabilísticamente estable lo denotaremos como $s \uparrow$.

□

Para definir la semántica de los grafos necesitamos introducir una nueva clase de transiciones, las transiciones probabilísticas, de la forma $s \rightarrow_q s'$. Así, en la Tabla 8.1 se muestran las reglas que definen la evolución de un grafo de estados dinámico probabilístico, donde $s = (n, C, t)$ es un estado del grafo, *act*, *adjust* y $Min_\tau(n, C)$ son exactamente las mismas que las definidas en el capítulo 4.

Las reglas G1 y G2 son similares a las que se presentaron en la Tabla 4.1 del capítulo 4, las cuales capturan, respectivamente, la evolución desde un estado probabilísticamente estable mediante la ejecución de alguna acción, y la evolución mediante el paso del tiempo.

La nueva regla G3 captura la evolución probabilística del grafo a partir de un estado probabilísticamente no estable.

G1) $$\frac{s = (n, C, t),\ s \downarrow,\ act(s, e)}{s \xrightarrow{e} (m, adjust(C, e, t), t)}$$

G2) $$\frac{s = (n, C, t),\ s \downarrow,\ t' \geq t,\quad t' - C(R_g) \leq min_\tau(n, C)}{s \underset{t'-t}{\longmapsto} (n, C, t')}$$

G3) $$\frac{s = (n, C, t),\ s \uparrow,\ n \longrightarrow_q n'}{s \longrightarrow_q (n', C, t)}$$

Tabla 8.1: Reglas de transición para los grafos probabilísticos

Definición 52 (Computación Probabilístico-temporizada $s_0 \overset{<w>}{\Longrightarrow}_{t,p} s$)
Sea $G = (NT, NP, ET, EP, \lambda, \gamma, n_0, clocks)$ un grafo de estados dinámico probabilístico, $w \in (Act \cup \mathbb{R}_0^+ \cup \{[q] \mid q \in (0, 1]\})^*$ una secuencia temporizada y probabilística (cadena de acciones, tiempos y probabilidades), $e \in Act$, $t \in \mathbb{R}_0^+$ y $q \in (0, 1]$.

1. Sea $s_0 = (n_0, C_0, 0)$ el estado inicial de G, y $s_0 \xrightarrow{e} s'$ una posible evolución en este grafo. Entonces $s_0 \overset{<e>}{\Longrightarrow}_{0,1} s'$ es una computación probabilístico-temporizada.

2. Sea $s_0 = (n_0, C_0, 0)$ el estado inicial de G y $s_0 \underset{t}{\longmapsto} s'$ una posible evolución de G. Entonces $s_0 \overset{<t>}{\Longrightarrow}_{t,1} s'$ es una computación probabilístico-temporizada.

3. Sea $s_0 = (n_0, C_0, 0)$ el estado inicial de G y $s_0 \longrightarrow_q s'$ una posible evolución de G. Entonces $s_0 \overset{<[q]>}{\Longrightarrow}_{0,q} s'$ es una computación probabilístico-temporizada.

4. Sea $s_0 \overset{<w>}{\Longrightarrow}_{t,p} s'$ una computación probabilístico-temporizada en G y $s' \xrightarrow{e} s''$ una posible evolución en G. Entonces $s_0 \overset{<w.e>}{\Longrightarrow}_{t,p} s''$ es una computación probabilístico-temporizada.

5. Sea $s_0 \overset{<w>}{\Longrightarrow}_{t,p} s'$ una computación probabilístico-temporizada en G y $s' \underset{t'}{\longmapsto} s''$ una posible evolución de G. Entonces $s_0 \overset{<w.t>}{\Longrightarrow}_{t+t',p} s''$ es una computación probabilístico-temporizada.

6. Sea $s_0 \stackrel{<w>}{\Longrightarrow}_{t,p} s'$ una computación probabilístico-temporizada en G y $s' \longrightarrow_q s''$ una posible evolución de G. Entonces $s_0 \stackrel{<w.[q]>}{\Longrightarrow}_{t,p\times q} s''$ es una computación probabilístico-temporizada.

□

8.2. Traducción de TPPAL a grafos probabilísticos

En esta sección presentaremos la construcción del grafo correspondiente a cada uno de los operadores del álgebra TPPAL. Ésta será una extensión natural de la construcción presentada para el lenguaje TPAL, y sólo será aplicable a los términos de RTPPAL.

8.2.1. Stop

El grafo para el proceso *stop* estará formado por un único nodo temporizado y un único reloj que será el reloj global. Así, la definición del grafo será:

$$G[\![stop]\!] = (\{stop\}, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, stop, \{R_g\})$$

8.2.2. Prefijo temporizado

Sea el proceso $T = a < t_1, t_2 >; T_1$ donde el grafo asociado con el proceso T_1 es $G[\![T_1]\!] = (NT_1, NP_1, ET_1, EP_1, \lambda_1, \gamma_1, n_{01}, clocks_1)$.

El grafo asociado al proceso T se obtiene a partir del grafo $G[\![T_1]\!]$ añadiendo un nuevo nodo temporizado y un nuevo arco temporizado. Este arco une el nuevo nodo con el nodo inicial de $G[\![T_1]\!]$. El conjunto de nodos y arcos probabilísticos no se verá modificado por lo que será el mismo de T_1. Así pues, el grafo para el proceso T será:

$$G[\![T]\!] = (NT,\ NP_1,\ ET,\ EP_1,\ \lambda,\ \gamma_1,\ P,\ clocks_1)$$

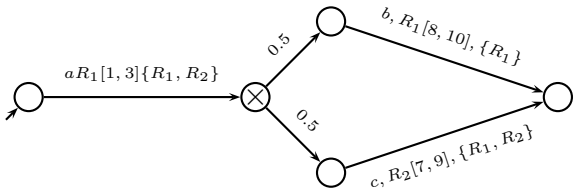
donde

$$NT = NT_1 \cup \{P\}$$
$$ET = ET_1 \cup \{P \xrightarrow{e} n_{01}\}$$
$$\lambda(\bar{e}) = \begin{cases} \lambda_1(\bar{e}) & \forall \bar{e} \in ET_1 \\ (a, Intg, clocks_1) & \text{si } \bar{e} = e \end{cases}$$

La función $Intg$, que define las restricciones temporales asociadas a esa transición, se define igual que en el capítulo 4.

El tamaño de $G[\![P]\!]$ es el tamaño de $G[\![P_1]\!]$ incrementado con un nodo temporizado y un arco temporizado.

Ejemplo 37 Sea el proceso $a < 1, 3 >; ([0,5]b < 8, 10 >; stop + [0,5]c < 7, 9 >; stop)$. El grafo obtenido para este proceso es el mostrado en la Figura 8.1.



$$clocks = \{R_1, R_2\}$$

Figura 8.1: Grafo correspondiente al proceso del ejemplo 37

□

8.2.3. Prefijo de acción urgente y operador wait

La construcción del grafo asociado a estos operadores es similar a la construcción dada para el prefijo temporizado, con la única diferencia de la etiqueta asignada al nuevo arco introducido. La definición de esta etiqueta es la misma que la realizada en el capítulo 4.

8.2.4. Elección externa

La construcción del grafo para el proceso $N = N_1 \Box N_2$ es la misma que la realizada en el capítulo 4, es decir:

$$G[\![N_1 \Box N_2]\!] = G[\![Unfold(N_1)]\!] \Box_G \; G[\![Unfold(N_2)]\!]$$

donde la función *Unfold* fue definida en el capítulo 4 (ver Def. 35). Dado que tanto N_1 como N_2 son procesos no-deterministas, obtenidos de acuerdo con la sintaxis definida en el capítulo 4, y que la operación *Unfold* fue definida para estos términos, no se hace necesario una extensión de la definición del operador *Unfold*.

Por tanto, en este caso debemos redefinir la operación $\Box_G$ de modo que actúe sobre dos grafos de estados dinámicos probabilísticos. Para ello, sean los procesos N_1 y N_2, cuyos grafos asociados son $G_i = (NT_i, NP_i, ET_i, EP_i, \lambda_i, \gamma_i, n_{0i}, clocks_i)$, $i = 1, 2$, los cuales cumplen la restricción impuesta de que los nodos iniciales no poseen arcos de entrada.

El grafo obtenido será:

$$G_1 \,\Box_G\, G_2 = (NT,\ NP,\ ET,\ EP,\ \lambda,\ \gamma,\ n_0,\ clocks)$$

donde:

$$\begin{array}{ll}
NT = & (NT_1 \cup NT_2 \cup \{N\}) \setminus \{n_{01}, n_{02}\} \\
n_0 = & N \\
ET = & ((ET_1 \cup ET_2) \setminus \\
 & (\{(n_{01}, n_1) \,|\, (n_{01}, n_1) \in ET_1\} \cup \{(n_{02}, n_2) \,|\, (n_{02}, n_2) \in ET_2\})) \cup \\
 & (\{(n_0, n_1) \,|\, (n_{01}, n_1) \in ET_1\} \cup \{(n_0, n_2) \,|\, (n_{02}, n_2) \in ET_2\}) \\
clocks = & clocks_1 \cup clocks_2 \\
R_g = & R_{g1} \text{ ó } R_{g2} \\
\lambda(e) = & (\bar{\lambda_1}(e),\ \bar{\lambda_2}(e),\ \bar{\lambda_3}(e) \cup \{R_g\}) \quad \forall e \in ET \\
NP = & NP_1 \cup NP_2 \\
EP = & EP_1 \cup EP_2 \\
\gamma(e) = & \begin{cases} \gamma_1(e) & \text{si } e \in EP_1 \\ \gamma_2(e) & \text{si } e \in EP_2 \end{cases}
\end{array}$$

$\bar{\lambda}_j$ (con $j = 1, 2, 3$) se define así:

$$\bar{\lambda_j}(e) = \begin{cases} \lambda_{ij}(e) & \forall e \in ET_i \cap ET \\ \lambda_{ij}(e_i) & \text{si } e = (n_0, n_i),\ \text{con } e_i = (n_{0i}, n_i) \end{cases}$$

siendo λ_{ij} la componente j de λ_i.

El nuevo grafo tiene $n_1 + n_2 - 1$ nodos y $e_1 + e_2$ arcos, siendo $n_1 = |NT_1| + |NP_1|$, $n_2 = |NT_2| + |NP_2|$, $e_1 = |ET_1| + |EP_1|$ y $e_2 = |ET_2| + |EP_2|$.

8.2.5. Elección probabilística

A diferencia de lo que ocurre con el operador de elección externa, en este caso no se hace necesario imponer la restricción de que los nodos iniciales de los grafos que intervienen en la elección no posean arcos entrantes. Ésto es debido a que la forma de construir el grafo no modifica el nodo destino de ninguno de los arcos de los grafos involucrados sino que se añaden nuevos arcos que unen el nuevo nodo introducido con los nodos iniciales de los grafos involucrados.

Por tanto no es necesario aplicar la función *Unfold* a los subtérminos de la elección.

Consideremos el proceso $P = \sum_{i \in I} q_i N_i$, donde cada proceso N_i posee su grafo asociado:

$$G[\![N_i]\!] = (NT_i, NP_i, ET_i, EP_i, \lambda_i, \gamma_i, n_{0i}, clocks_i)$$

Para construir el grafo asociado al proceso P deberemos añadir un nuevo nodo probabilístico y tantos arcos probabilísticos como procesos N_i intervengan en la elección. Estos arcos tendrán como nodo origen el nuevo nodo introducido, y como nodo destino el nodo inicial n_{0i} de cada uno de los grafos, y estarán etiquetados con la probabilidad q_i asociada a ese proceso.

Formalmente, el grafo asociado al proceso P será:

$$G[\![P]\!] = (NT, NP, ET, EP, \lambda, \gamma, n_0, clocks)$$

donde

$$\begin{aligned}
NT &= \bigcup_{i \in I} NT_i \\
NP &= \bigcup_{i \in I} NP_i \cup \{P\} \\
ET &= \bigcup_{i \in I} ET_i \\
EP &= \bigcup_{i \in I} EP_i \bigcup_{i \in I} (P \longrightarrow_{q_i} n_{0i}) \\
\lambda(e) &= (\bar{\lambda}_1(e),\ \bar{\lambda}_2(e),\ \bar{\lambda}_3(e) \cup \{R_g\}) \quad \forall\, e \in ET \\
\gamma(e) &= \begin{cases} \gamma_i(e) & \text{si } e \in EP_i \\ q_i & \text{si } e \in EP \setminus EP_i \end{cases} \\
n_0 &= P \\
clocks &= \bigcup_{i \in I} clocks_i
\end{aligned}$$

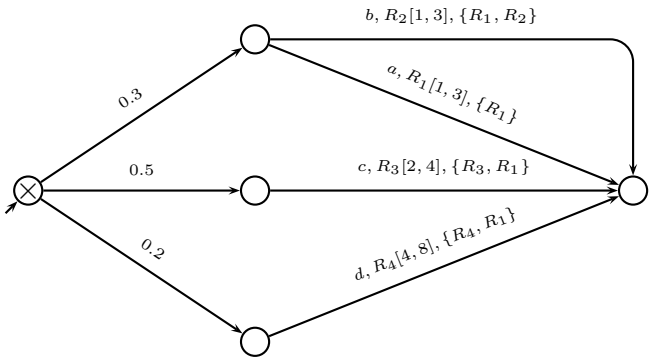
Al igual que en la elección externa, el reloj global del nuevo grafo será el de alguno de los grafos $G[\![N_i]\!]$.

El número de nodos del nuevo grafo será $|NT_1| + |NT_2| + |NP_1| + |NP_2| + 1$, y el número de arcos del nuevo grafo será $|ET_1| + |ET_2| + |EP_1| + |EP_2| + n$, donde el valor de n será el número de términos que intervienen en la elección.

Ejemplo 38 Consideremos el proceso siguiente:

$$[0{,}3](a < 1, 3 >; stop \Box b < 1, 3 >; stop) + [0{,}5]c < 2, 4 >; stop + [0{,}2]d < 4, 8 >; stop$$

El grafo asociado que obtendremos aplicando la definición anterior será el mostrado en la Figura 8.2.



$$clocks = \{R_1, R_2, R_3, R_4\}\ R_g = R_1$$

Figura 8.2: Grafo correspondiente al proceso del ejemplo 38

□

8.2.6. Composición paralela

Sea el proceso $T = T_1 \|_{\{A\}} T_2$, donde los procesos T_1 y T_2 tienen asociados los grafos $G[\![T_i]\!] = (NT_i, NP_i, ET_i, EP_i, \lambda_i, \gamma_i, n_{0i}, clocks_i)$, para $i = 1, 2$.

El grafo para el proceso T será:

$$G[\![T]\!] = (NT, NP, ET, EP, \lambda, \gamma, n_0, clocks)$$

el cual se obtiene mediante el producto cartesiano de los grafos asociados a los procesos T_1 y T_2. En este nuevo grafo tendremos que:

- El conjunto de relojes *clocks* contendrá los relojes de ambos grafos, y uno nuevo que será introducido y que será el reloj global del nuevo grafo. Este

reloj se incluirá en la etiqueta de todos los arcos temporizados del grafo. El conjunto de relojes asociado al grafo será por tanto:

$$clocks = clocks_1 \cup clocks_2 \cup \{R_g\},\ \text{t.q. } R_g \notin clocks_1 \cup clocks_2$$

- El conjunto de nodos temporizados contendrá todos aquellos nodos obtenidos a partir del producto cartesiano de los nodos temporizados, mientras que el conjunto de nodos probabilísticos contendrá los obtenidos a partir de uno temporizado y uno probabilístico o bien a partir de dos probabilísticos. El nodo inicial del nuevo grafo será el nodo obtenido a partir de los nodos iniciales de ambos grafos, pudiendo ser temporizado o probabilístico.

$$\begin{aligned} NT &= NT_1 \times NT_2 \\ NP &= (NT_1 \times NP_2) \cup (NP_1 \times NT_2) \cup (NP_1 \times NP_2) \\ n_0 &= (n_{01}, n_{02}) \end{aligned}$$

- El conjunto de arcos temporizados contendrá aquellos arcos de ambos grafos cuyo nodo origen en el nuevo grafo sea un nodo temporizado. Los arcos de ambos grafos que estén etiquetados con la misma acción de sincronización se fusionarán en un único arco. Aquellos arcos temporizados cuyo nodo origen en el nuevo grafo vaya a ser un nodo probabilístico serán eliminados. Así, la definición de los conjuntos de arcos temporizados es la siguiente:

$$ET = ET_{11} \cup ET_{12} \cup ET_{13}$$

donde:

$$\begin{aligned} ET_{11} = &\ \{((n_1,n_2),(n_1',n_2)) \,|\, (n_1,n_2) \in NT,\ n_1 \xrightarrow{e_1}_1 n_1',\ \lambda_{11}(e_1) \notin A\} \\ ET_{12} = &\ \{((n_1,n_2),(n_1,n_2')) \,|\, (n_1,n_2) \in NT,\ n_2 \xrightarrow{e_2}_2 n_2',\ \lambda_{21}(e_2) \notin A\} \\ ET_{13} = &\ \{((n_1,n_2),(n_1',n_2')) \,|\, (n_1,n_2) \in NT,\ n_1 \xrightarrow{e_1}_1 n_1',\ n_2 \xrightarrow{e_2}_2 n_2', \\ &\ \lambda_{11}(e_1) = \lambda_{21}(e_2) \in A\} \end{aligned}$$

El conjunto de arcos probabilísticos se define como:

$$EP = EP_{11} \cup EP_{12} \cup EP_{13}$$

donde:

$$\begin{aligned} EP_{11} = &\ \{((n_1,n_2),(n_1',n_2)) \,|\, n_2 \in NT_2,\ (n_1,n_1') \in EP_1\} \\ EP_{12} = &\ \{((n_1,n_2),(n_1,n_2')) \,|\, n_1 \in NT_1,\ (n_2,n_2') \in EP_2\} \\ EP_{13} = &\ \{((n_1,n_2),(n_1',n_2')) \,|\ (n_1,n_1') \in EP_1,\ (n_2,n_2') \in EP_2\} \end{aligned}$$

- Las funciones de etiquetado para los arcos serán las siguientes:

$$\lambda(e) = \begin{cases} (\lambda_{11}(e_1), \lambda_{12}(e_1), \lambda_{13}(e_1) \cup \{R_g\}) & \text{para } e_1 \in ET_1,\ e \in ET_{11} \\ (\lambda_{21}(e_2), \lambda_{22}(e_2), \lambda_{23}(e_2) \cup \{R_g\}) & \text{para } e_2 \in ET_2,\ e \in ET_{12} \\ (a, \lambda_{12}(e_1) \cup \lambda_{22}(e_2), \lambda_{13}(e_1) \cup \lambda_{23}(e_2) \cup \{R_g\}) & \text{para } e \in ET_{13}, a = \lambda_{11}(e_1) \end{cases}$$

$$\gamma(e) = \begin{cases} \gamma_1(e_1) & \text{para } e_1 \in EP_1,\ e \in EP_{11} \\ \gamma_2(e_2) & \text{para } e_2 \in EP_2,\ e \in EP_{12} \\ \gamma_1(e_1) \times \gamma_2(e_2) & \text{para } e \in EP_{13} \end{cases}$$

El tamaño del grafo obtenido será, en el peor de los casos, igual al producto de los tamaños de los grafos, reduciéndose el mismo dependiendo de las acciones de sincronización que existan.

Ejemplo 39 Consideremos el proceso siguiente:

$$a < 1,2 >; ([0,3]c < 1,2 >; stop + [0,7]b < 0,2 >; stop) \|_{\{b\}}$$
$$d < 1,2 >; ([0,4]e < 1,3 >; stop + [0,6]b < 0,3 >; stop)$$

El grafo asociado que obtendremos aplicando la definición será el mostrado en la Figura 8.3.

□

8.2.7. Ocultamiento

Consideremos ahora $T = T_1 \backslash a$, donde T_1 tiene asociado el grafo $G[\![T_1]\!] = (NT_1, NP_1, ET_1, EP_1, \lambda_1, \gamma_1, n_{01}, clocks_1)$.

El grafo asociado al proceso T será:

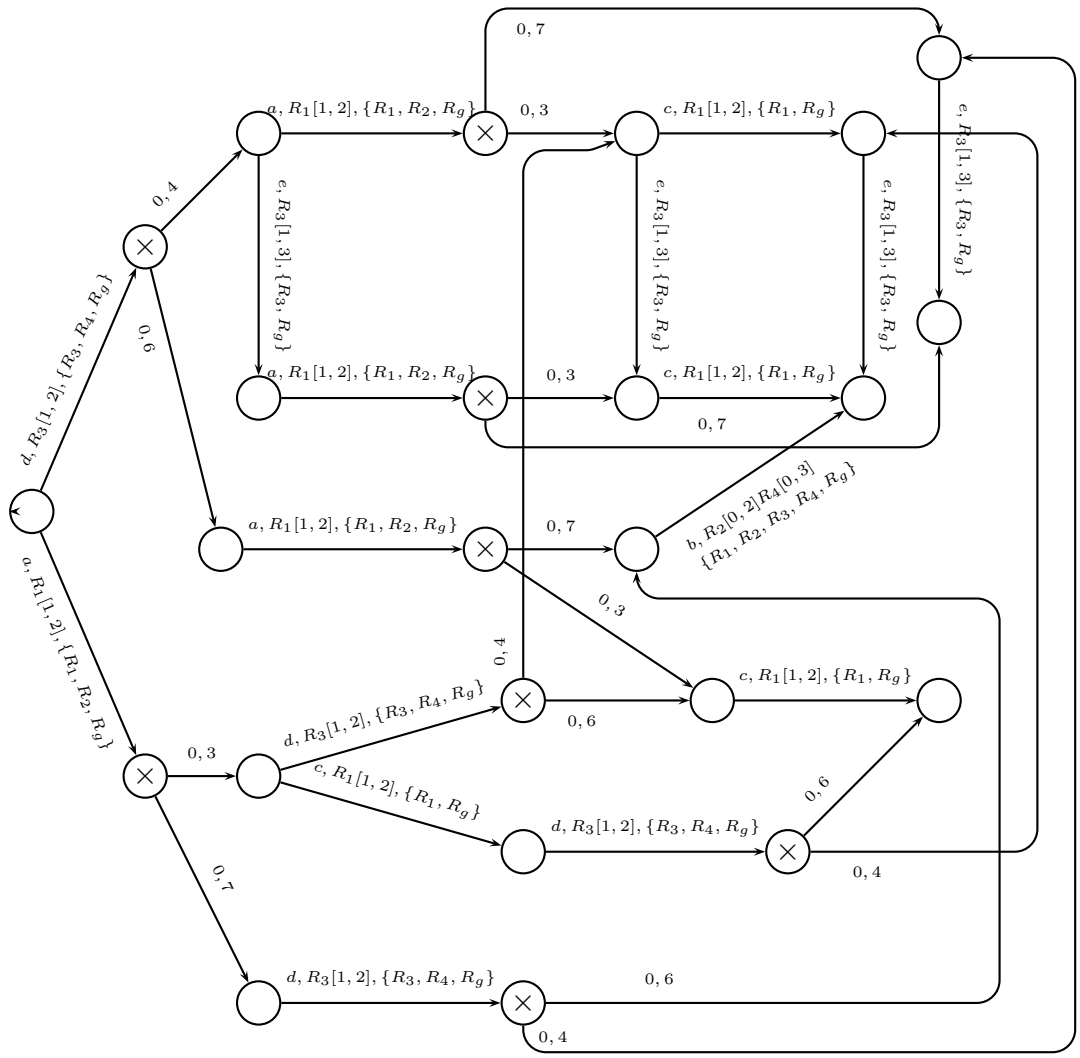
$$G[\![T]\!] = (NT_1 \backslash a, NP_1 \backslash a, ET_1, EP_1, \lambda, \gamma_1, n_{01}, clocks_1)$$

donde la función de etiquetado para los arcos será:

$$\lambda(e) = \begin{cases} (\tau, \lambda_{12}(e), \lambda_{13}(e)) & \forall e \in ET_1,\ \ \lambda_{11}(e) = a \\ \lambda_1(e) & \text{en caso contrario} \end{cases}$$

$NT_1 \backslash a$ y $NP_1 \backslash a$ representan los términos de NT_1 y NP_1 a los que se les ha añadido el subtérmino $\backslash a$.

Como puede observarse en la definición, al igual que se hacía para los grafos temporizados, la construcción del grafo asociado al proceso $T \backslash a$, se hace cambiando



$$clocks = \{R_1, R_2, , R_3, R_4, R_g\}$$

Figura 8.3: Grafo correspondiente al proceso del ejemplo 39

las etiquetas de las transiciones temporizadas etiquetadas con la acción a por la etiqueta τ.

El tamaño del grafo obtenido será exactamente el mismo que el del grafo $G[\![P_1]\!]$, ya que solamente se modifican las etiquetas de los arcos.

8.2.8. Identificador X

Al igual que ocurría en los grafos de estados dinámicos, dado que trabajamos con términos cerrados, parece que en principio no necesitamos asociar grafos a los identificadores. Ahora bien, dado que los identificadores se utilizan para la definición de la recursión, para definir correctamente el grafo del operador recursión nos apoyaremos de nuevo en estos grafos $G[\![X]\!]$, construidos como en el capítulo 4.

8.2.9. Recursión

Sea $T = \mu X.T_1$, donde el grafo construido para el proceso T_1 es $G[\![T_1]\!] = (NT_1,\ NP_1,\ ET_1,\ EP_1,\ \lambda_1,\ \gamma_1,\ n_{01}, clocks_1)$.

El grafo asociado al proceso $T = \mu X.T_1$ será:

$$G[\![T]\!] = (NT,\ NP_1,\ ET,\ EP,\ \lambda,\ \gamma,\ n_{01},\ clocks_1)$$

donde el conjunto de nodos temporizados será el conjunto de nodos temporizados de $G[\![T_1]\!]$ al que se le han retirado los nodos etiquetados con X, y donde cada ocurrencia de X en los términos que etiquetan los nodos es reemplazada por $\mu X.T_1$.

$$NT = NT_1 \setminus \{n \in NT_1 \mid term(n) = X\}$$

El conjunto de arcos se mantiene, modificando únicamente el nodo destino de aquellos arcos cuyo nodo destino era uno de los eliminados, pasando a ser el nodo inicial de $G[\![T_1]\!]$.

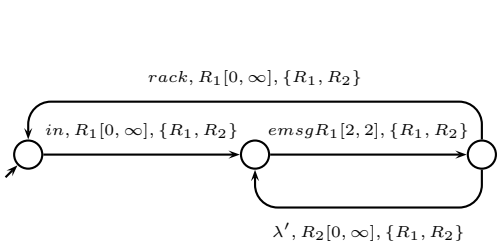
$$\begin{aligned} ET = \ & ET_1 \cup \{(n, n_{01}) \mid (n,m) \in ET_1 \ \wedge\ term(m) = X\} \setminus \\ & \{(n,m) \in ET \mid term(m) = X\} \\ EP = \ & EP_1 \cup \{(n, n_{01}) \mid (n,m) \in EP_1 \ \wedge\ term(m) = X\} \setminus \\ & \{(n,m) \in EP_1 \mid term(m) = X\} \end{aligned}$$

La función de etiquetado de los arcos se verá modificada, de modo que, para los arcos temporizados cuyo nodo destino se ha modificado, debe cambiarse el conjunto de relojes que sincronizan con el global. El nuevo conjunto para estos arcos será $clocks_1$. Así, la función de etiquetado será:

$$\lambda(e) = \begin{cases} \lambda_1(e) & \text{si } e \in ET_1 \\ (\lambda_{11}(e), \lambda_{21}(e), clocks_1) & \text{si } e = (n, n_{01}) \ \wedge (n,m) \in ET_1, term(m) = X \end{cases}$$

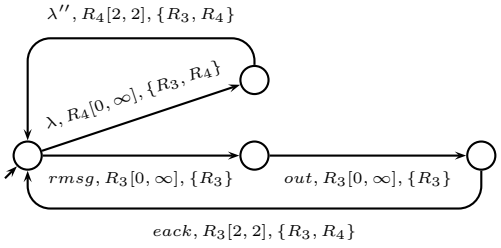
Ejemplo 40 Consideremos de nuevo el protocolo AUY, cuya especificación se mostró en el capítulo anterior. Aplicando la construcción anterior para cada uno de los procesos, para el emisor obtendremos el grafo mostrado en la Figura 8.4(a), para el receptor obtendremos el grafo de la Figura 8.4(b) y para los dos canales los mostrados en las Figuras 8.4(c) y 8.4(d) respectivamente.

El grafo para el sistema completo se obtendrá mediante el producto cartesiano de estos cuatro grafos, cuyo tamaño no nos permite mostrarlo aquí.



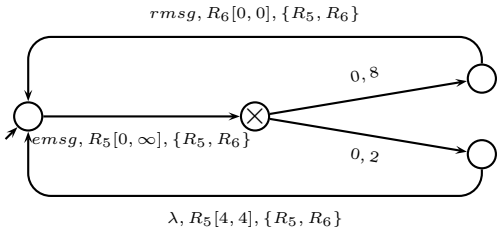
$clocks = \{R_1, R_2\}\ R_g = R_1$

(a) Grafo para el proceso emisor



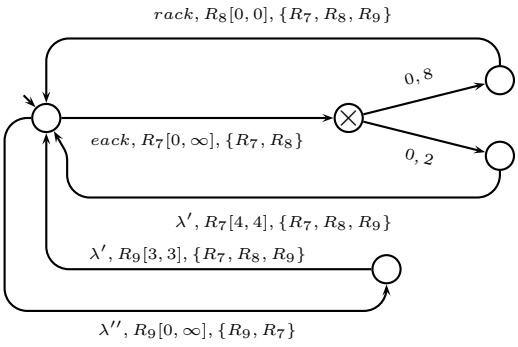
$clocks = \{R_3, R_4\}\ R_g = R_3$

(b) Grafo para el proceso receptor



$clocks = \{R_5, R_6\}\ R_g = R_5$

(c) Grafo para el canal C1



$clocks = \{R_7, R_8, R_9\}\ R_g = R_9$

(d) Grafo para el canal C2

Figura 8.4: Grafos para los procesos del protocolo AUY

□

8.3. Equivalencia

Antes de establecer la equivalencia entre los grafos de estados dinámicos probabilísticos y el álgebra debemos redefinir el conjunto de componentes secuenciales y la función *time_evol* que se definieron en el capítulo 4 para incluir el nuevo operador de elección probabilística.

Definición 53 (Componentes secuenciales)
Definimos el conjunto de componentes secuenciales con la siguiente sintaxis:

$$S ::= stop \mid a < t_1, t_2 >; T \mid \tau; T \mid wait(t); T \mid S \,\square\, S \mid \sum_{i \in I} [q_i] S_i \mid S \parallel_A \mid {}_A\!\parallel S \mid S \backslash a$$

con $T \in TPPAL$.

Entonces, definimos el conjunto de componentes secuenciales de un proceso regular cerrado T mediante la función:

$$dex \,:\, RTPPAL \longrightarrow \mathcal{P}_F(S)$$

la cual está definida como sigue:

$$\begin{aligned}
&dex(stop) = \{stop\} \\
&dex(a < t_1, t_2 >; P) = \{a < t_1, t_2 >; P\} \\
&dex(\tau; P) = \{\tau; P\} \\
&dex(wait(t); P) = \{wait(t); P\} \\
&dex(P_1 \square P_2) = dex(P_1) \;\square\; dex(P_2) \\
&dex(\sum_{i \in I} [q_i] N_i) = \sum_{i \in I} dex[q_i](N_i) \\
&dex(P_1 \parallel_A P_2) = dex(P_1) \parallel_A \;\cup\; {}_A\!\parallel dex(P_2) \\
&dex(P \backslash a) = dex(P) \backslash a \\
&dex(\mu X.P) = dex(P\{\mu X.P/X\})
\end{aligned}$$

□

Una vez definida la descomposición en componentes secuenciales, podemos extender la definición de *"time_evol"*.

Definición 54 (time_evol)
Dados dos procesos regulares cualesquiera P y Q de TPPAL. Diremos que $P \in time\_evol(Q)$ si y sólo si existe una biyección $\varphi : dex(Q) \longrightarrow dex(P)$ tal que:

$$\forall\, S_Q \in dex(Q)\; \exists\, r \in \mathbb{R}_0^+,\; old(S_Q, r) = \varphi(S_Q)$$

donde la función *old* está definida como sigue:

$$
\begin{aligned}
&old(stop, r) = stop \\
&old(\tau; P, r) = \begin{cases} stop & \text{si } r > 0 \\ \tau; P & \text{si } r = 0 \end{cases} \\
&old(a < t_1, t_2 >; P, r) = \begin{cases} stop & \text{si } r > t_2 \\ a < t_1 \ \dot{-} \ r, t_2 - r >; P & \text{si } r \leq t_2 \end{cases} \\
&old(wait(t); P, r) = \begin{cases} stop & \text{si } r > t \\ wait(t - r); P & \text{si } r \leq t \end{cases} \\
&old(S_1 \Box S_2, r) = old(S_1, r) \Box old(S_2, r) \\
&old(\sum_{i \in I} [q_i] N_i, r) = \begin{cases} stop & \text{si } r > 0 \\ \sum_{i \in I} [q_i] N_i & \text{si } r = 0 \end{cases} \\
&old(S \|_A, r) = old(S, r) \|_A \\
&old({}_A \| S, r) = {}_A \| old(S, r) \\
&old(S \backslash a, r) = old(S, r) \backslash a
\end{aligned}
$$

□

Teorema 6

Dados dos procesos regulares cualesquiera T, T' de *RTPPAL*, si tenemos que:

$$T \stackrel{<s>}{\Longrightarrow}_{t,p} T'$$

entonces existe un término $\bar{T}' \in TPPAL$ y un nodo $n_{\bar{T}'} \in G[\![T]\!]$ tal que $T' \in time\_evol(\bar{T}')$, y existe un estado $(n_{\bar{T}'}, C, t)$ en $G[\![T]\!]$, tal que:

$$(n_0, C_0, 0) \stackrel{<s>}{\Longrightarrow}_{t,p} (n_{\bar{T}'}, C, t)$$

Demostración:

La demostración de este teorema la realizaremos por inducción estructural sobre los operadores del lenguaje, y para cada uno de los operadores procederemos por inducción sobre la longitud de la secuencia s utilizada para alcanzar el proceso T'.

- Caso base: *stop*
  El razonamiento es idéntico al realizado en el capítulo 4.

- Prefijos, wait y elección externa
  Para estos operadores el razonamiento es idéntico al realizado en el capítulo 4 para los mismos.

- Elección probabilística: $P = \sum_{i \in I} [q_i] N_i$

  - Caso base ($|s| = 1$)
    La única regla que puede aplicarse en este caso es $P \longrightarrow_{q_i} T' = N_i$.

    En $G[\![T]\!]$ tendremos para cada $i \in I$ un arco desde el nodo inicial al nodo inicial del grafo $G[\![N_i]\!]$. Esta transición nos permitirá alcanzar con probabilidad q_i el estado $(n_{N_i}, C, 0)$, tal que $N_i \in time\_evol(N_i)$.

  - Caso general ($|s| > 1$)
    Consideremos que la secuencia realizada es $P \longrightarrow_{q_i} N_i \stackrel{<s_1>}{\Longrightarrow}_{t,q} T'$. Al igual que en el caso anterior, en $[\![T]\!]$ podemos alcanzar el nodo N_i en el instante 0 tras realizar una transición probabilística.

    Además, aplicando la hipótesis de inducción sobre el proceso N_i, tenemos que existe un estado $(n_{\bar{T}'}, C, t)$ alcanzable en ese grafo con probabilidad q, y que cumple que $T' \in time\_evol(\bar{T}')$.

    Por tanto podemos concluir que existe un estado $(n_{\bar{T}'}, C, t)$ alcanzable con probabilidad $q_i \times p$ en $G[\![P]\!]$, que cumple $T' \in time\_evol(\bar{T}')$.

- Composición paralela: $T = T_1 \|_A T_2$

  - Caso base ($|s| = 1$)
    Distinguiremos varios casos dependiendo de si los procesos T_i son temporizados o probabilísticos.

    - Caso b.1 ($T = N_1 \|_A N_2$)
      Basta con repetir un razonamiento similar al realizado en el capítulo 4 para este operador.

    - Caso b.2 ($T = P_1 \|_A N_2$)
      El proceso T evoluciona mediante una transición probabilística $T \longrightarrow_{q_1} T_1'$. Por la sintaxis del proceso T sabemos que esa transición proviene de una transición $P_1 \longrightarrow_{q_i} N_1'$.
      En $G[\![P_1]\!]$ tendremos una transición etiquetada con q_1 desde el nodo raíz hasta el nodo $n_{N_1'}$, que por la hipótesis de inducción puede ser ejecutada alcanzando ese nodo en el instante 0, de modo que $N_1' \in time\_evol(N_1')$.
      En $G[\![T]\!]$ tendremos un arco etiquetado como el anterior que va desde el nodo raíz (n_{01}, n_{02}) hasta el nodo $(n_{N_1'}, n_{02})$ correspondiente al proceso $N_1' \|_A N_2$; de modo que podremos alcanzar el estado

$((n_{N_1'}, n_{02}), C, 0)$ con probabilidad q_1, cumpliendose además $N_1' \|_A N_2 \in$ *time_evol*$(N_1' \|_A N_2)$.

- Caso b.3 ($T = N_1 \|_A P_2$)
  Este caso es el simétrico del anterior.
- Caso b.4 ($T = P_1 \|_A P_2$)
  En este caso, la evolución del proceso T será $T \longrightarrow_q N_1 \|_A N_2$, y por la sintaxis del mismo sabemos que cada uno de los procesos involucrados en la composición paralela, P_1 y P_2 puede evolucionar de acuerdo con las siguientes transiciones $P_1 \longrightarrow_{q_1} N_1$ y $P_2 \longrightarrow_{q_2} N_2$, respectivamente, y $q = q_1 \times q_2$.
  Si aplicamos la hipótesis de inducción a ambos procesos tendremos que en sus respectivos grafos existe un arco desde n_{P_i} hasta n_{N_i}, y además en este caso en concreto $N_i \in$ *time_evol*(N_i). Si trasladamos esto al grafo de $T = P_1 \|_A P_2$ tendremos que existe un arco desde el nodo (n_{P_1}, n_{P_2}) hasta el nodo (n_{N_1}, n_{N_2}), que puede ser ejecutado y nos lleva al estado $((n_{N_1}, n_{N_2}), C_0, 0)$ con probabilidad $q_1 \times q_2$. Además, $N_1 \|_A N_2 \in$ *time_evol*$(N_1 \|_A N_2)$.

- Caso general ($|s| > 1$)
  Para realizar la generalización para secuencias s tal que $|s| > 1$ partiremos la secuencia s en dos subsecuencias: s_1 que contiene las transiciones (acciones, retrasos y decisiones probabilísticas) de P_1 y s_2 que contiene las transiciones de P_2. La intersección de estas dos subsecuencias contendrá las acciones de sincronización. Esta división de la secuencia s se hace un poco más compleja que en el caso de los grafos de estados dinámicos ya que cada una de las transiciones probabilísticas deberá ser dividida en la parte que corresponde al proceso P_1 y en la parte que corresponde a P_2.

  De acuerdo con esta división tendremos $P_1 \stackrel{<s_1>}{\Longrightarrow}_{t_1,q_1} T_1'$ y $P_2 \stackrel{<s_2>}{\Longrightarrow}_{t_2,q_2} T_2'$, siendo $T' = T_1' \|_A T_2'$.

  Aplicando la hipótesis de inducción para cada s_i tendremos que ésta es realizable en el grafo $G[\![P_i]\!]$. A partir de esta consideración podemos concluir que la secuencia total s es posible en $G[\![P]\!]$, ya que el conjunto de relojes de $G[\![P_1]\!]$ es distinto del de $G[\![P_2]\!]$, y en su evolución por separado no interfieren unos en otros manteniendo los no modificados su "hora local".

Cuando se produce el paso del tiempo ambos procesos permiten el paso del tiempo, y por lo tanto el grafo obtenido en su composición paralela lo permite también.

Cuando se produce una transición probabilística en algunas de las secuencias, en la secuencia s podrá realizarse también, y si se produce la transición en ambas secuencias, entonces en la nueva secuencia se realizará una transición probabilística con una probabilidad igual al producto de las probabilidades de las dos transiciones consideradas.

El reloj global ha sido introducido en todos los nodos por lo que su actualización es correcta.

Así pues, tenemos que tras la ejecución de la secuencia s estaremos en el nodo $(n_{\bar{T_1'}}, n_{\bar{T_2'}})$ con probabilidad $q_1 \times q_2$, y por la aplicación de la hipótesis de inducción que hicimos sobre las subsecuencias tenemos que $T_1' \in time\_evol(\bar{T_1'})$ y $T_2' \in time\_evol(\bar{t_2'})$.

Para finalizar comprobaremos que $T' = T_1' \|_A T_2' \in time\_evol(\bar{T_1'} \|_A \bar{T_2'})$, para lo cual debemos definir una biyección:

$$\varphi : dex(\bar{T_1'} \|_A \bar{T_2'}) \longrightarrow dex(T_1' \|_A T_2')$$

Como $T_i' \in time\_evol(\bar{T_i'})$ $i = 1, 2$ tenemos:

$$\exists \varphi_i : dex(\bar{T_i'}) \longrightarrow T_1'$$

que cumple las condiciones establecidas en la definición 54

Por la definición de componente secuencial tenemos

$$\varphi(S\|_A) = \varphi_1(S)\|_A$$
$$\varphi({}_A\|S) = {}_A\|\varphi_2(S)$$

Así, $\forall\, S\|_A \in dex(\bar{T_1'} \|_A \bar{T_2'})$ tenemos que $S \in dex(\bar{T_1'})$ cumpliendose $\exists\, r$ tal que $old(S, r) = \varphi_1(S)$. De ahí, y de acuerdo con la definición de old tenemos:

$$old(S\|_A, r) = old(S, r)\|_A = \varphi_1(S)\|_A$$

De forma similar procederemos para ${}_A\|S \in dex(\bar{T_1'} \|_A \bar{T_2'})$

- Ocultamiento $T = T_1 \backslash a$
  Recordemos que $G[\![T]\!]$ se obtiene cambiando las etiquetas de todos los arcos etiquetados con a por τ en $G[\![T_1]\!]$.

Aplicando la hipótesis de inducción y la regla R9c es fácil concluir que este cambio no afecta a otras acciones que eran ejecutables en T, es decir, no se deshabilitan acciones en $G[\![T]\!]$ por el carácter urgente de las nuevas τ que aparecen.

Entonces, si tenemos la secuencia $T_1\backslash a \stackrel{<s>}{\Longrightarrow}_{t,q} T'\backslash a$ tendremos la secuencia equivalente en T_1, $T_1 \stackrel{<s_1>}{\Longrightarrow}_{t,q} T'$, y usando la hipótesis de inducción obtenemos que existe un estado $(n_{\bar{T}'_1}, C, t)$ alcanzable en $G[\![T_1]\!]$ de modo que $T'_1 \in \mathit{time\_evol}(\bar{T}'_1)$. Entonces la misma secuencia s_1 puede ser ejecutada en $G[\![T]\!]$ donde las apariciones de a han sido sustituidas por τ.

- Recursión $T = \mu\, X.Q$
  - Caso base ($|s| = 1$)
    - Caso b.1 ($T \stackrel{a}{\longrightarrow} T'$)
      Si el proceso T puede realizar la acción a es porque el proceso $Q\{\mu X.Q/X\}$ la puede ejecutar también. Entonces podemos aplicar la hipótesis de inducción sobre el proceso Q con lo que obtendremos que existe un nodo $n_{\bar{T}'}$ alcanzable al ejecutar la acción a en tiempo 0 por lo que $T' \in \mathit{time\_evol}(T')$.
      Si el nodo alcanzado en $G[\![Q]\!]$ estaba etiquetado con X, en $G[\![T]\!]$ este nodo habrá desaparecido, y todos los arcos que llegaban a él ahora llegan al nodo inicial de Q. Por lo tanto, el nodo que se alcanzará en $G[\![T]\!]$ será el nodo inicial.
      Si el nodo estaba etiquetado con otro proceso distinto de X, entonces éste aparecerá en $G[\![T]\!]$, y será alcanzable en tiempo 0, según la semántica del grafo.
    - caso b.2 ($T \stackrel{t}{\longmapsto} T'$)
      En este caso T podrá dejar pasar el tiempo si $Q\{\mu X.Q\,/\,X\}$ lo permite. Aplicando la hipótesis de inducción sobre Q tendremos que alcanzaremos un nodo $n_{\bar{T}'}$, que es el inicial de Q y que cumple que $T' \in \mathit{time\_evol}(\bar{T}')$.
    - caso b.3 ($T \longrightarrow_p T'$)
      El proceso T evolucionará mediante una elección probabilística si el proceso $Q\{\mu X.Q\,/\,X\}$ lo permite. Aplicando la hipótesis de inducción sobre Q tendremos que alcanzaremos un nodo $n_{\bar{T}'}$, en tiempo 0 y con probabilidad p; además por hipótesis de inducción $T' \in$

$time\_evol(\bar{T}')$.

- Caso general $|s| > 1$
  De nuevo troceamos la secuencia s de la forma siguiente: $s = s_1\ s_2\ s_3\ \ldots\ s_n$ con $P \stackrel{s_1}{\Longrightarrow}_{t_1,q_1} P_1 \stackrel{s_2}{\Longrightarrow}_{t_2,q_2} P_2 \Longrightarrow \ldots P_{n-1} \stackrel{s_n}{\Longrightarrow}_{t_n,q_n} P_n$, siendo $t = t_1 + t_2 + \cdots + t_n$, $q = q_1 \times q_2 \times \cdots \times q_n$ y cada $P_i = \mu X.Q$.
  Esta división hace que todas las secuencias s_i se inicien en el nodo inicial de T. Aplicando la hipótesis de inducción podemos asegurar que cada una de esas trazas s_i es reproducible en el grafo de T partiendo del nodo inicial de T y con todos los relojes sincronizados (Lema 1 que puede ser fácilmente extendido).

□

Al igual que para TPAL, podemos observar que el inverso no es cierto en general, porque no todo $P' \in time\_evol(\bar{P}')$ puede ser alcanzado usando la semántica operacional. En su lugar tenemos el siguiente resultado para la inversa.

Teorema 7
Dado un proceso regular cualquiera T de *RTPPAL* y su grafo asociado $G[\![T]\!]$.

Si $(n_0, C_0, 0) \stackrel{<s>}{\Longrightarrow}_{t,p} (n_{\bar{T}'}, C, t)$ entonces existe un proceso regular $T' \in RTPPAL$ tal que $T \stackrel{<s>}{\Longrightarrow}_{t,p} T'$, donde $T' \in time\_evol(\bar{T}')$.

Demostración: Al igual que para el teorema anterior, procederemos por inducción estructural sobre los operadores del lenguaje, y para cada uno de ellos por inducción sobre la longitud de la secuencia s.

- Caso base: *stop*
  El razonamiento es idéntico al realizado en el capítulo 4.

- Prefijos, wait y elección externa
  Estos procesos no permiten realizar ninguna transición probabilística, por lo que para ellos el razonamiento que debemos realizar es idéntico al realizado en el capítulo 4.

- Elección probabilística $P = \sum_{i \in I} [q_i]N_i$
  En el grafo asociado a este proceso, para cada $i \in I$ existirá un arco probabilístico que parte del nodo inicial y que estará etiquetado con q_i. Procederemos por inducción sobre la longitud de s.

- Caso base ($|s| = 1$)
  En $[\![P]\!]$ se realiza la transición

  $$(n_0, C_0, 0) \longrightarrow_{q_i} (n_{N_i}, C_0, 0)$$

  La transición equivalente en el álgebra se obtiene por aplicación de la regla $P1$, la cual nos lleva al proceso N_i, y se cumple trivialmente que $N_i \in \mathit{time\_evol}(N_i)$.

- Caso general ($|s| > 1$)
  Consideremos que la secuencia ejecutada en el grafo es la siguiente

  $$(n_0, C_0, 0) \longrightarrow_{q_i} (n_{N_i}, C_0, 0) \stackrel{<s_1>}{\Longrightarrow}_{t,p} (n_{\bar{T}'}, C, t)$$

  Tras la realización de una elección probabilística nos encontraremos en el nodo n_{N_i} asociado al proceso N_i en tiempo 0, con probabilidad q_i. A partir de este nodo se podrá realizar la subtraza s_1.

  Aplicando la hipótesis de inducción tendremos que existe un proceso N_i' tal que $N_i \stackrel{<s_1>}{\Longrightarrow}_{t,p} T_i'$ y $N_i' \in \mathit{time\_evol}(\bar{T}')$. Por tanto, podemos concluir que $P = \sum_{i \in I} [q_i]N_i \stackrel{<s_1>}{\Longrightarrow}_{t,p \times q_i} T' = N_i'$.

▪ Composición paralela: $T = T_1 \|_A T_2$

- Caso base ($|s| = 1$)
  Podemos distinguir los casos siguientes:

  ◦ Caso b.1 ($T = N_1 \|_A N_2$)
  La demostración para este caso es idéntica a la realizada en el capítulo 4.

  ◦ Caso b.2 ($T = P_1 \|_A N_2$)
  En este caso la transición realizada en $G[\![T]\!]$ será de la forma:

  $$(n_0, C_0, 0) \longrightarrow_p (n_{N_1 \|_A N_2}, C_0, 0)$$

  Esta transición es posible porque en $G[\![P_1]\!]$ tenemos la transición:

  $$(n_{01}, C, 0) \longrightarrow_p (n_{N_1}, C, 0)$$

  Aplicando la hipótesis de inducción sobre este grafo tenemos que $P_1 \longrightarrow_p N_1$, con $N_1 \in \mathit{time\_evol}(N_1)$, y aplicando la regla $P2a$ tenemos:

  $$T \longrightarrow_p N_1 \|_A N_2$$

cumpliéndose trivialmente:

$$N_1 \|_A N_2 \in \mathit{time\_evol}(N_1 \|_A N_2)$$

- Caso b.3 ($T = N_1 \|_A P_2$)
  Es el caso simétrico al anterior.
- Caso b.4 ($T = P_1 \|_A P_2$)
  En este caso, si el grafo $G[\![T]\!]$ puede ejecutar una transición de la forma:

  $$(n_0, C_0, 0) \longrightarrow_p (n_{N_1 \|_A N_2}, C_0, 0)$$

  es porque en el grafo $G[\![P_1]\!]$ se puede ejecutar una transición de la forma:

  $$(n_{01}, C_{1,0}, 0) \longrightarrow_{p'} (n_{N_1}, C_{1,0}, 0)$$

  y en $G[\![P_2]\!]$ se puede ejecutar una transición:

  $$(n_{02}, C_{2,0}, 0) \longrightarrow_{p''} (n_{N_2}, C_{2,0}, 0)$$

  Aplicando la hipótesis de inducción sobre ambos grafos tendremos que $P_1 \longrightarrow_{p'} N_1$ y $P_2 \longrightarrow_{p''} N_2$, con $N_1 \in \mathit{time\_evol}(N_1)$ y $N_2 \in \mathit{time\_evol}(N_2)$.
  Aplicando la regla $P2c$ tendremos que:

  $$T \longrightarrow_{p' \times p''} N_1 \|_A N_2$$

  y además $N_1 \|_A N_2 \in \mathit{time\_evol}(N_1 \|_A N_2)$.

- Caso general ($|s| > 1$)
  Al igual que hicimos para demostrar el teorema 6, dividiremos la secuencia s en dos subsecuencias, s_1 compuesta por las transiciones ejecutadas en $G[\![P_1]\!]$ y s_2 compuesta por las transiciones ejecutadas por $G[\![P_2]\!]$.
  Aplicando la hipótesis de inducción a ambas tendremos que $P_1 \stackrel{<s_1>}{\Longrightarrow}_{t_1,q_1} T_1'$ y $P_2 \stackrel{<s_2>}{\Longrightarrow}_{t_2,q_2} T_2'$, con $T_1' \in \mathit{time\_evol}(\bar{T}_1')$ y $T_2' \in \mathit{time\_evol}(\bar{T}_2')$.
  Si consideramos la composición paralela de P_1 y P_2 podremos intercalar las acciones de s_1 y de s_2 en el orden en el que aparecían en s y por tanto tendremos que $P \stackrel{<s>}{\Longrightarrow}_{t,p} T_1' \|_A T_2'$. Comprobemos que además se cumple

  $$T_1' \|_A T_2' \in \mathit{time\_evol}(\bar{T}_1' \|_A \bar{T}_2')$$

  para lo cual deberemos definir una biyección:

  $$\varphi : \mathit{dex}(\bar{T}_1' \|_A \bar{T}_2') \longrightarrow \mathit{dex}(T_1' \|_A T_2')$$

Como $T'_i \in time\_evol(\bar{T}'_i)$ $i = 1, 2$ tenemos:

$$\exists \varphi_i : dex(\bar{T}'_i) \longrightarrow T'_1$$

que cumple las condiciones establecidas en la definición 54.

Por la definición de componente secuencial tenemos

$$\varphi(S\|_A) = \varphi_1(S)\|_A$$
$$\varphi({}_A\|S) = {}_A\|\varphi_2(S)$$

Así, $\forall\, S\|_A \in dex(\bar{T}'_1\|_A\bar{T}'_2)$ tenemos que $S \in dex(\bar{T}'_1)$ cumpliendose $\exists\, r$ tal que $old(S, r) = \varphi_1(S)$. De ahí, y de acuerdo con la definición de *old* tenemos:

$$old(S\|_A, r) = old(S, r)\|_A = \varphi_1(S)\|_A$$

De forma similar procederemos para ${}_A\|S \in dex(\bar{T}'_1\|_A\bar{T}'_2)$.

- Ocultamiento $T = T_1\backslash a$

  Consideremos la secuencia de transiciones s en $G[\![T]\!]$ que permite alcanzar el estado $(n_{\bar{T}'\backslash a}, C, t)$. Entonces en $G[\![T_1]\!]$ podremos ejecutar también la secuencia correspondiente s' , obtenida "destapando"las acciones "a" ocultadas.

  Aplicando la hipótesis de inducción obtendremos que existe un proceso T' tal que $T_1 \stackrel{<s'>}{\Longrightarrow}_{t,q} T'$ y $T' \in time\_evol(\bar{T}')$. Aplicando la semántica operacional tendremos $T_1\backslash a \stackrel{<s>}{\Longrightarrow}_{t,p} T'\backslash a$ y $T'\backslash a \in time\_evol(T'\backslash a)$.

- Recursión $P = \mu X.Q$

  De nuevo se trocea la computación $(n_0, C_0, 0) \stackrel{<s>}{\Longrightarrow}_{t,q} (n_{\bar{P}'}, C, t)$, dividiendo s en $s = s_1\ s_2\ \ \ldots\ s_n$, tales que

  $$(n_0, C_{i-1}, t_{i-1}) \stackrel{<s_i>}{\Longrightarrow}_{\Delta t_i, q_i} (n_0, C_i, t_i)$$

  con $t_i = t_{i-1} + \Delta t_i$, $C = C_n$, $t = t_n$, y $q = q_1 \times \cdots \times q_n$. Es decir, cada s_i nos lleva desde el nodo inicial de nuevo al nodo inicial en tiempo Δt_i.

  Cada una de estas secuencias s_i serán también secuencias de $G[\![Q\{\mu X.Q/X\}]\!]$, por lo que aplicando la hipótesis de inducción para cada una de ellas tendremos que $Q \stackrel{<s_i>}{\Longrightarrow}_{t_i,q_i} Q_i$ con $Q_i \in time\_evol(X)$, salvo la última, en la cual $Q_n \in time\_evol(\bar{P}')$.

  Así pues tendremos que $P \stackrel{<s_1>}{\Longrightarrow}_{t_1,q_i} P_1 \stackrel{<s_2>}{\Longrightarrow}_{t_2,q_2} P_2 \Longrightarrow \ldots \stackrel{<s_n>}{\Longrightarrow}_{t_n,q_n} P_n = P'$ por lo que podemos concluir que $P \stackrel{<s>}{\Longrightarrow}_{t,q} P_n = P'$.

□

Capítulo 9

TPAL. Una herramienta de apoyo al análisis y diseño de sistemas concurrentes

Junto al objetivo de establecer una relación entre diferentes modelos temporizados y probabilísticos utilizados para la especificación de sistemas concurrentes, otro de los objetivos de esta tesis era el que esta traducción se hiciese de forma automática.

De cara a poder cumplir este objetivo, en 1997 se empezó a desarrollar la herramienta TPAL, cuyo desarrollo está intimamente ligado al desarrollo de esta tesis, de modo que la evolución en la implementación de la misma ha estado muy marcada por la evolución de la tesis.

El desarrollo de esta herramienta es un proyecto ambicioso, que pretende la construcción de una herramienta que permita el análisis y evaluación de sistemas concurrentes, así como que sirva de puente entre diversas herramientas ya existentes como UPPAAL [BLPY95] y LOLA (Lotos Laboratory)[DIT95].

La versión actual de la herramienta TPAL se encuentra disponible en las direcciónes *ftp://penelope.info-ab.uclm.es/TPAL*, *http://www.info-ab.uclm.es/fmc/tpaltool/tpaltool.htm*, en dos versiones, una para sistemas Linux bajo plataforma PC y otra para sistemas Solaris bajo plataforma Sun Sparc. Los requerimientos necesarios para su correcto funcionamiento son escasos, siendo necesario como software adicional las librerías gráficas Xforms. Además, es conveniente disponer de gran can-

tidad de memoria RAM, dado que el tamaño de los grafos generados puede llegar a ser grande, y éstos residen en memoria principal.

9.1. Principales características

La herramienta TPAL, al encontrarse en fase de desarrollo, se encuentra sometida a continuos cambios, con el fin de incrementar su funcionalidad y facilitar su utilización.

Las principales características de la versión actual de TPAL son las siguientes:

- Permite la creación y edición de especificaciones en el lenguaje algebraico temporizado y probabilístico TPPAL. Estas especificaciones son creadas dentro de proyectos, los cuales permiten descomponer la especificación de sistemas complejos en varias componentes, cada una de los cuales agrupa varios procesos.
- Análisis sintáctico y semántico de las especificaciones.
- Visualización de los árboles sintácticos en modo texto.
- Traducción de las especificaciones temporizadas a grafos de estados dinámicos.
- Reducción simple de los grafos generados.
- Visualización de los grafos en modo texto o gráfico.
- Simulación, bien automática o bien manual, del sistema, usando los grafos obtenidos.
- Tras una simulación, permite salvar la traza seguida para posteriormente ser recuperada, o ser estudiada detenidamente.
- Traducción de las especificaciones a Redes de Petri con arcos Temporizados.
- Visualización en modo texto de las redes de Petri obtenidas.

En las siguientes secciones de este capítulo procederemos a describir más detenidamente las diferentes características de la herramienta, y los objetivos que pretendemos cubrir en el futuro. Para ilustrar cada una de las distintas características de esta herramienta utilizaremos el siguiente ejemplo, en el cual se modela el sistema de transporte en un río, cuyo funcionamiento se detalla a continuación.

Ejemplo 41 (Barcas)
En un río existen dos barcas que permiten cruzarlo en ambos sentidos, cada una de las cuales únicamente admite un pasajero. Inicialmente cada una de las barcas se encuentra en una de las orillas del río. Cuando llega un pasajero a una orilla, si hay alguna barca, sube y se traslada a la orilla opuesta. Si por el contrario no hay ninguna barca, los pasajeros disponen de un botón en cada orilla del río, el cual se encuentra asociado a una luz en el extremo contrario que se ilumina al pulsar el botón, y permanece así hasta que el operador de la barca la apague, desde la orilla donde está la luz. Ésto lo hará cuando vaya a salir para atender esa petición. Puede ocurrir que haya varios pasajeros esperando, y puede que en la orilla opuesta no haya nadie, de modo que los pasajeros pulsan el botón cada cierto tiempo, hasta ser atendidos.

□

9.2. Análisis sintáctico y semántico

9.2.1. Sintaxis de los proyectos

Como se comentaba en la descripción general de las características de la herramienta, las especificaciones se organizan en proyectos, cada uno de los cuales está compuesto de varios procesos, los cuales son susceptibles de ser agrupados en componentes. Cada uno de los proyectos vendrá definido por un fichero con extensión "*.prj*", cuya estructura es la mostrada en la Figura 9.1, en el cual se pueden distinguir los siguientes apartados.

En primer lugar aparece el apartado *PROCESSES*, en el cual se indica el nombre de todos los procesos que intervienen en el proyecto.

Una vez indicados los procesos que intervienen en el sistema, se definirán cada una de las componentes en las que se descompone el sistema, las cuales estarán formadas por algunos de los procesos enumerados, así como por otras de las componentes ya definidas. La definición de cada una de estas componentes vendrá delimitada por las palabras reservadas *Component* y *End_component*. Para cada una de las componentes, además de los procesos que la componen, se especifica el conjunto de acciones de sincronización de los mismos.

Para finalizar el fichero, aparece la sección correspondiente al proyecto comple-

```
-- List of processes involved:
--     EXAMPLE:    Processes P1, P2;
Processes  ;
-- List of Componentes involved in the project specification
-- This list could be empty:
--     EXAMPLE:    Component C1
Component P123
   -- Enumeration of processes in the component (at least one):
   Processes  ;
   -- List of actions that must be syncrhonized for this component
   -- (can be empty)
   Sync     ;
End_component
-- Other components, include new component specifications:
-- Project specification, the name must coincide with that
-- of this file, without the extension:
Project ;
   -- Enumeration of processes in the component (at least one):
   Processes  ;
   -- List of synchronized actions
   Sync  ;
   -- List of hidden actions
   Hidden ;
End_project
```

Figura 9.1: Fichero de proyecto

to, dentro de la sección *Project*, cuyo nombre asignado debe coincidir con el nombre del fichero del proyecto sin incluir la extensión. Al igual que para cada una de las componentes, para el proyecto se indicarán los procesos o componentes que intervienen y el conjunto de acciones de sincronización. Además se indicará el conjunto de acciones que son ocultadas.

En el fichero podemos incluir los comentarios que consideremos pertinentes, para lo cual se precederán con doble guión.

En la Figura 9.2 puede verse el fichero de proyecto para el sistema descrito en el ejemplo 41.

Tras la creación de un nuevo proyecto, se habrá creado un directorio, cuyo nombre es el mismo del proyecto pero con la extensión "*.dir*". En este directorio deberán colocarse los ficheros correspondientes a las especificaciones de cada uno de los procesos. El nombre de cada uno de los ficheros deberá coincidir con el nombre del

```
Processes Ba1, Ba2, C1, C2, LUZ1, LUZ2, PASAJEROAB, PASAJEROBA;
Component Barcas
   Processes Ba1, Ba2 ;
   Sync     ;
End_component
Component Control
   Processes C1, C2;
   Sync ;
End_Component
Component Luces
   Processes LUZ1, LUZ2;
   Sync ;
End_component
Component ControlBarcas
   Processes Barcas, Control;
   Sync initab,initba,finab,finba;
End_component
Component Pasajeros
   Processes PASAJEROAB, PASAJEROBA;
   Sync;
End_component
Component BarcasControlesLuces
   Processes ControlBarcas, Luces;
   Syn apagada1,apagada2,ok1,ok2,nook1,nook2,desconecta1,desconecta2;
End_component
Project barcas3;
   Processes BarcasControlesLuces, Pasajeros;
   Sync  pulsa1,pulsa2,subeab,subeba,bajaa,bajab;
   Hidden initab,initba,finab,finba,apagada1,ok1,nook1,pulsa1,
          desconecta1,pulsa2,desconecta2,apagada2,ok2,nook2;
End_project
```

Figura 9.2: Fichero de proyecto para el ejemplo de las barcas

proceso que especifica.

9.2.2. Sintaxis de los procesos.

En la especificación de cada uno de los procesos que intervienen en el sistema tendremos un proceso principal y varios procesos auxiliares.

La especificación del proceso principal se ajusta a la sintaxis mostrada en la Figura 9.3, donde:

```
SPECIFICATION <id_proceso> [<lista_acciones>];
BEHAVIOUR [PROBABILISTIC | NON_DETERMINISTIC]
<cuerpo_proceso>
[WHERE <lista_procesos>]
ENDSPEC
```

Figura 9.3: Sintaxis del proceso principal

id_proceso es el identificador de la especificación en cuestión.

lista_acciones es una lista que contiene todas aquellas acciones utilizadas dentro de la especificación.

lista_procesos es la especificación de todos aquellos procesos auxiliares utilizados en la especificación. Éstos serán especificados de acuerdo a la sintaxis mostrada en la Figura 9.4.

```
PROCESS <id_proceso> [<lista_acciones>];
BEHAVIOUR [PROBABILISTIC | NON_DETERMINISTIC]
<cuerpo_proceso>
ENDPROC
```

Figura 9.4: Sintaxis de proceso auxiliar

cuerpo_proceso Es la definición del comportamiento de ese proceso. Esta definición se hará de acuerdo a la sintaxis del lenguaje algebraico TPPAL, cuya definición formal presentamos en el capítulo 7.

De cara a un mejor tratamiento automático se han introducido unas ligeras modificaciones en la sintaxis como son:

- La acción interna se representa con i en vez de τ.
- El valor ∞ se representa por $*$.
- El operador de paralelo se representa por $[A]$, siendo A el conjunto de acciones de sincronización.
- El operador "elección externa" se representa por "*or*".

Para cada uno de los procesos, opcionalmente puede indicarse el tipo de proceso (PROBABILISTIC o NON_DETERMINISTIC). En caso de no ser indicado, la herramienta presenta la facilidad de poder determinar dicho tipo en la mayoría de los casos.

Retomando de nuevo el sistema de cruce del río, descrito en el ejemplo 41, las especificaciones para cada uno de los procesos involucrados en el sistema son las que se muestran en las figuras siguientes. Concretamente, la Figura 9.5 muestra la especificación de un pasajero que viaja de la orilla B a la orilla A, mientras que en la Figura 9.6 se muestra la del pasajero en sentido contrario. En la Figura 9.7 se muestra la especificación de los sistemas de control y en la Figura 9.8, la especificación de los sistemas de luces. Para finalizar, en las Figuras 9.9 y 9.10 se muestran las especificaciones del comportamiento de las dos barcas.

```
SPECIFICATION PASAJEROBA [pasea,ok1,apagada1,nook1,pulsa1,desconecta1,
                          initab,initba,vab,vba,finab,finba,subeab,subeba,
                          bajaa,bajab,pulsa2,desconecta2,apagada2,ok2,nook2];
BEHAVIOUR
   rec X. pasea<50,*>; (rec Y. ( (subeba<0,0>; bajaa<0,*>; PASAJEROAB1)
                                 or (wait(2); pulsa1<0,0>; Y) ) )

WHERE
PROCESS PASAJEROAB1[pasea,ok1,apagada1,nook1,pulsa1,desconecta1,
                    initab,initba,vab,vba,finab,finba,subeab,subeba,
                    bajaa,bajab,pulsa2,desconecta2,apagada2,ok2,nook2];
BEHAVIOUR
    rec X. pasea<5,*>; (rec Y. ( (subeab<0,0>; bajab<0,*>; PASAJEROBA)
                                 or (wait(2); pulsa1<0,0>; Y) ) )
ENDPROC
ENDSPEC
```

Figura 9.5: Especificación de un pasajero que viaja de la orilla B a la A

9.2.3. Análisis sintáctico y semántico

La comprobación sobre la corrección sintáctica y semántica se realiza en dos fases.

En la primera fase se comprueba que las especificaciones se ajustan a la sintaxis del lenguaje; obteniendo como resultado una primera versión del árbol sintáctico, la

```
SPECIFICATION PASAJEROAB [pasea,ok1,apagada1,nook1,pulsa1,desconecta1,
                          initab,initba,vab,vba,finab,finba,subeab,subeba,
                          bajaa,bajab,pulsa2,desconecta2,apagada2,ok2,nook2];
BEHAVIOUR
    rec X. pasea<5,10>; (rec Y. ( (subeab<0,0>; bajab<0,*>; PASAJEROBA1)
                                  or  (wait(2); pulsa1<0,0>; Y) ) )

WHERE
PROCESS PASAJEROBA1[pasea,ok1,apagada1,nook1,pulsa1,desconecta1,
                    initab,initba,vab,vba,finab,finba,subeab,subeba,
                    bajaa,bajab,pulsa2,desconecta2,apagada2,ok2,nook2];
BEHAVIOUR
   rec X. pasea<7,10>; (rec Y. ( (subeba<0,0>; bajaa<0,*>; PASAJEROAB)
                                  or (wait(2); pulsa1<0,0>; Y) ) )
ENDPROC
ENDSPEC
```

Figura 9.6: Especificación de pasajero que viaja de la orilla A a la orilla B

```
SPECIFICATION C2 [pasea,ok1,apagada1,nook1,pulsa1,desconecta1,
                  initab,initba,vab,vba,finab,finba,subeab,subeba,
                  bajaa,bajab,pulsa2,desconecta2,apagada2,ok2,nook2];
BEHAVIOUR
   rec X. initba<0,*>; finba<0,*>;X
ENDSPEC

SPECIFICATION C1 [pasea,ok1,apagada1,nook1,pulsa1,desconecta1,
                  initab,initba,vab,vba,finab,finba,subeab,subeba,
                  bajaa,bajab,pulsa2,desconecta2,apagada2,ok2,nook2];
BEHAVIOUR
    rec X. initab<0,*>; finab<0,*>;X
ENDSPEC
```

Figura 9.7: Especificación de controles

cual será incompleta, ya que en el mismo faltará cierta información semántica, en concreto, los tipos de los procesos e identificadores.

En la segunda fase se determinará (siempre que sea posible) esta información. En los casos en los que no puede detectarse el tipo de los procesos o identificadores de forma automática, se generará un error, y el usuario tendrá que indicar explícitamente el tipo (PROBABILISTIC o NON_DETERMINISTIC).

```
SPECIFICATION LUZ1 [pasea,ok1,apagada1,nook1,pulsa1,desconecta1,
                    initab,initba,vab,vba,finab,finba,subeab,subeba,
                    bajaa,bajab,pulsa2,desconecta2,apagada2,ok2,nook2];
BEHAVIOUR
  rec X. ( (apagada1<0,*>; ok1<0,*>; X)
           or (desconecta1<0,*>; LUZ1) or (pulsa1<0,*>; LUZ11))
WHERE
PROCESS LUZ11[pasea,ok1,apagada1,nook1,pulsa1,desconecta1,
              initab,initba,vab,vba,finab,finba,subeab,subeba,
              bajaa,bajab,pulsa2,desconecta2,apagada2,ok2,nook2];
BEHAVIOUR
   rec X. ( (apagada1<0,*>; nook1<0,*>; X)
            or (pulsa1<0,*>; X) or (desconecta1<0,*>; LUZ1) )
ENDPROC
ENDSPEC

SPECIFICATION LUZ2 [pasea,ok1,apagada1,nook1,pulsa1,desconecta1,
                    initab,initba,vab,vba,finab,finba,subeab,subeba,
                    bajaa,bajab,pulsa2,desconecta2,apagada2,ok2,nook2];
BEHAVIOUR
  rec X. ( ( apagada2<0,*>; ok2<0,*>; X)
   or ( desconecta2<0,*>; X) or (pulsa2<0,*>; LUZ22) )
WHERE
PROCESS LUZ22[pasea,ok1,apagada1,nook1,pulsa1,desconecta1,
              initab,initba,vab,vba,finab,finba,subeab,subeba,
              bajaa,bajab,pulsa2,desconecta2,apagada2,ok2,nook2];
BEHAVIOUR
   rec X. ( ( apagada2<0,*>; nook2<0,*>; X)
            or ( pulsa2<0,*>; X) or ( desconecta2<0,*>; LUZ2))
ENDPROC
ENDSPEC
```

Figura 9.8: Especificación de luces

Tras finalizar el análisis sintáctico y semántico tendremos un árbol sintáctico por cada uno de los procesos de la especificación. Estos árboles tendrán en el nodo raíz la información del operador principal del proceso, y cada uno de sus hijos será el subárbol correspondiente a cada uno de los operandos involucrados en ese operador.

Entre la información almacenada para cada operador tendremos el tipo del proceso (NO_DETERMINISTA o PROBABILÍSTICO), la descripción del mismo (operador) y los argumentos que lo definen.

```
SPECIFICATION Ba1 [pasea,ok1,apagada1,nook1,pulsa1,desconecta1,
                   initab,initba,vab,vba,finab,finba,subeab,subeba,
                   bajaa,bajab,pulsa2,desconecta2,apagada2,ok2,nook2];
BEHAVIOUR
   rec X.( ( subeab<0,*>; desconecta1<0,*>; initab<0,*>;
             vab<10,15>; finab<0,0>; bajab<0,0>; B22 )
          or ( wait(1); apagada1<0,0>;
     ( ( ok1<0,*>; X) or
               ( nook1<0,*>; desconecta1<0,*>; initab<0,*>;
                 vab<10,15>; finab<0,0>; B22 ) ) ) )
WHERE
PROCESS B22[pasea,ok1,apagada1,nook1,pulsa1,desconecta1,
            initab,initba,vab,vba,finab,finba,subeab,subeba,
            bajaa,bajab,pulsa2,desconecta2,apagada2,ok2,nook2];
BEHAVIOUR
   rec X.( ( subeba<0,*>; desconecta2<0,*>; initba<0,*>;
             vba<10,15>; finba<0,0>; bajaa<0,0>; B1  )
           or ( wait(1); apagada2<0,0>;
              ( ( ok2<0,0>; X ) or
               ( nook2<0,*>; desconecta2<0,*>; initba<0,*>;
                 vba<10,15>; finba<0,0>; B1 ) ) ) )
ENDPROC
ENDSPEC
```

Figura 9.9: Especificación de la barca 1

En la Figura 9.11 podemos observar la definición de las estructuras de datos mediante las cuales se construye el grafo.

Si nos centramos de nuevo en el problema de las barcas tenemos que el grafo generado para el proceso PASAJEROBA se muestra en la Figura 9.12, siendo la subfigura 9.12(a) la vista generada por la herramienta; mientras que en la subfigura 9.12(b) se muestra una representación gráfica del mismo.

9.3. Grafos

Una de las posibilidades que ofrece la herramienta, a partir de los árboles sintácticos, es la construcción de los grafos de estados dinámicos, actualmente sólo para procesos temporizados, pero esperamos finalizar en breve plazo la implementación

```
SPECIFICATION Ba2 [pasea,ok1,apagada1,nook1,pulsa1,desconecta1,
                   initab,initba,vab,vba,finab,finba,subeab,subeba,
                   bajaa,bajab,pulsa2,desconecta2,apagada2,ok2,nook2];
BEHAVIOUR
   rec X.( ( subeba<0,*>; desconecta2<0,*>; initba<0,*>;
             vba<10,15>; finba<0,0>; bajaa<0,0>; B11 )
           or  ( wait(1); apagada2<0,0>;
               (( ok2<0,0>; X ) or
               ( nook2<0,*>; desconecta2<0,*>; initba<0,*>;
                 vba<10,15>; finba<0,0>; B11 ) ) ) )
WHERE

PROCESS B11[pasea,ok1,apagada1,nook1,pulsa1,desconecta1,
            initab,initba,vab,vba,finab,finba,subeab,subeba,
            bajaa,bajab,pulsa2,desconecta2,apagada2,ok2,nook2];
BEHAVIOUR
   rec X.( ( subeab<0,*>; desconecta1<0,*>; initab<0,*>;
             vab<10,15>; finab<0,0>; bajab<0,0>; B2 )
          or ( wait(1); apagada1<0,0>;
             ( ( ok1<0,*>; X) or
               ( nook1<0,*>; desconecta1<0,*>; initab<0,*>;
                 vab<10,15>; finab<0,0>; B2 ) ) )  )
ENDPROC
ENDSPEC
```

Figura 9.10: Especificación de la barca 2

para procesos probabilísticos.

Esta traducción se ajusta a la definición formal de los grafos realizada en el capítulo 4.

De cara a optimizar la gestión de la memoria de la máquina sobre la que se ejecuta la herramienta, y con el propósito de paliar en alguna medida el problema de la explosión de estados, en vez de construir un único grafo para todo el sistema, se crea un grafo modular, el cual está compuesto por un grafo para cada uno de los procesos de que conste la especificación.

En esta definición modular, además de los grafos es necesario mantener cierta información del sistema completo, que nos permita relacionar todos los grafos de forma correcta, principalmente durante la simulación. Entre esta información de coordinación tenemos las acciones que sincronizan, las acciones ocultadas y el reloj

```
struct NODO {
  char Tipo;             /* tipo de nodo (P:prob.; N:no-determinista) */
  char Desc;             /* Descripción de operador de mas alto nivel */
  struct Argumentos Arg; /* Argumentos */
  struct NODO *Hijos[MAXPALT];  /* Nodos hijos */
  int numhijos;                 /* Numero de hijos */
};

struct Argumentos{
/* stop, interna urgente, or: No tienen */
/* Prefijo: */
     char Nombre[TAMANOMBRES];   /* Nombre de la acción */
     unsigned int t1, t2;        /* Tiempos de prefijo temporizado */
/* Wait: */
/*     unsigned int t1;           Tiempo del wait */
/* Paralelo: */
     struct LISTAR c_sinc;       /* Conjunto de acciones de sincronización*/
/* Ocultamiento */
/*     char Nombre [TAMANOMBRE];   Acción ocultada */
/* Identificador */
/*     char Nombre [TAMANOMBRE];   Nombre del identificador */
/* Recursión */
/*    char Nombre [TAMANOMBRE];   Nombre del identificador */
/* Elección Probabilística */
    float p [MAXPALT];          /* Probabilidades*/
};
```

Figura 9.11: Información del arbol sintáctico

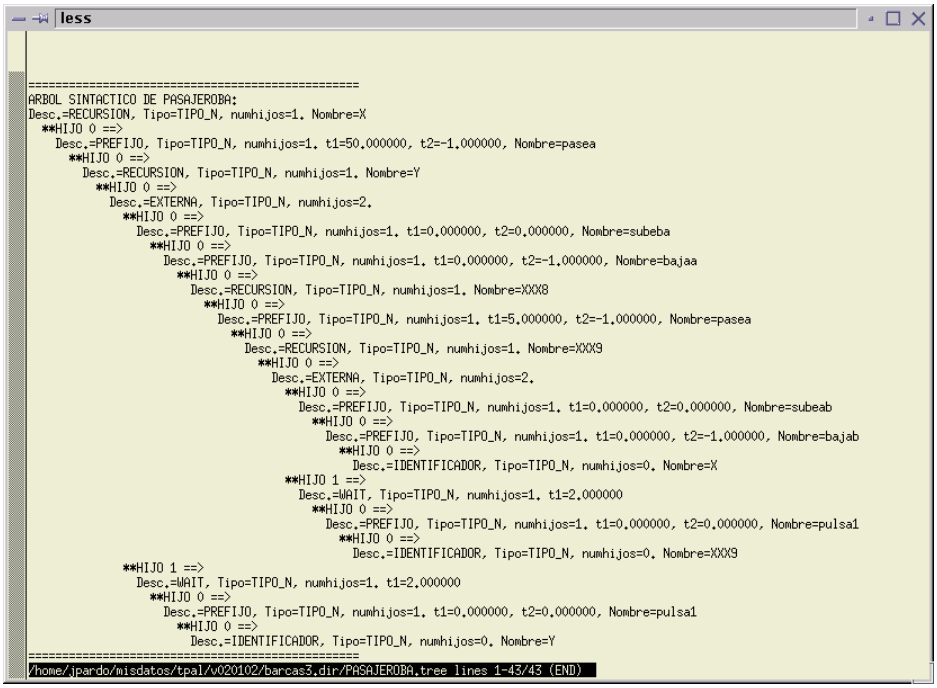
global de todo el sistema.

9.3.1. Browser y simulación

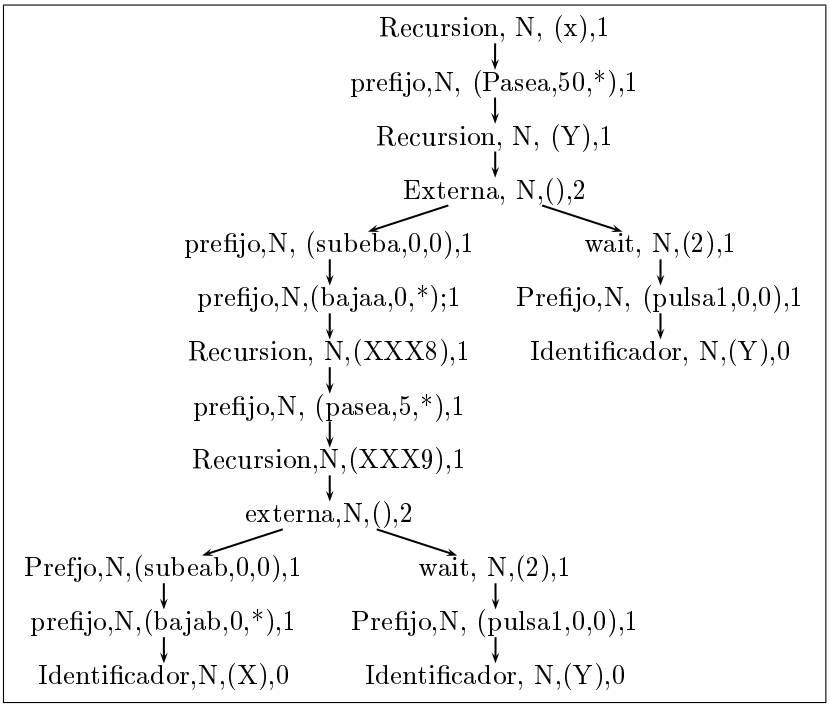
Ambas funciones aparecen unidas, de modo que una vez visualizado el grafo, puede realizarse la simulación.

En la visualización de los grafos aparecerán diversas ventanas, cada una de las cuales muestra el grafo correspondiente a uno de los procesos del sistema. En la visualización de un grafo podemos observar nodos de 3 colores:

Rojo.- Solamente existirá un nodo pintado de este color, que será el nodo activo en ese momento. Inicialmente será el nodo etiquetado como “1” y no será mo-



(a) Vista textual mostrada por la herramienta



(b) Vista gráfica

Figura 9.12: Árbol sintáctico generado para el proceso PASAJEROBA

dificado mientras que no se realice una simulación.

Amarillo.- Aparecerán pintados de este color aquellos nodos para los cuales, por falta de espacio en la ventana, no han podido ser dibujados todos sus nodos adyacentes.

Blanco.- El resto de nodos.

En esta visualización, pueden configurarse algunos parámetros, como son el tamaño de los nodos y sus etiquetas, así como la posibilidad de que las etiquetas de las transiciones sean visibles o estén ocultas.

En la Figura 9.13 podemos ver las ventanas generadas por la herramienta para mostrar el conjunto de grafos obtenidos para el problema de las barcas.

Simulación

A partir de los grafos generados, y sobre su visualización gráfica, podemos realizar una simulación de la ejecución del sistema.

Para mostrar la información generada por la simulación, y poder controlar la misma, junto a las ventanas que muestran los grafos, aparecerá la ventana de simulación, la cual puede verse en la Figura 9.14.

En la parte superior aparecerán los valores de cada uno de los relojes, mientras que en la parte inferior irá mostrándose la traza que se va ejecutando.

Además pueden configurarse diversos parámetros de la simulación, entre los cuales podemos destacar el tipo de simulación, si intentarán evitarse bloqueos por caducidad de acciones, la velocidad de la simulación, etc. Para ello se utilizará la ventana de configuración de la simulación, que se muestra en la Figura 9.15.

Si se selecciona la *simulación automática*, el propio sistema se encarga de ir seleccionando las transiciones que van siendo ejecutadas, de forma aleatoria de entre las habilitadas en cada momento. Si se ha seleccionado la opción "*Avoid time locks*" la selección de la acción a ejecutar se realizará de forma que intente evitarse el bloqueo del sistema por la caducidad de las acciones.

Si se elige la opción de simulación manual, será el usuario de la herramienta el que controle toda la evolución del sistema. Para indicar el paso del tiempo, el usuario deberá pulsar sobre el botón *Delay* de la ventana de simulación (ver Figura 9.14),

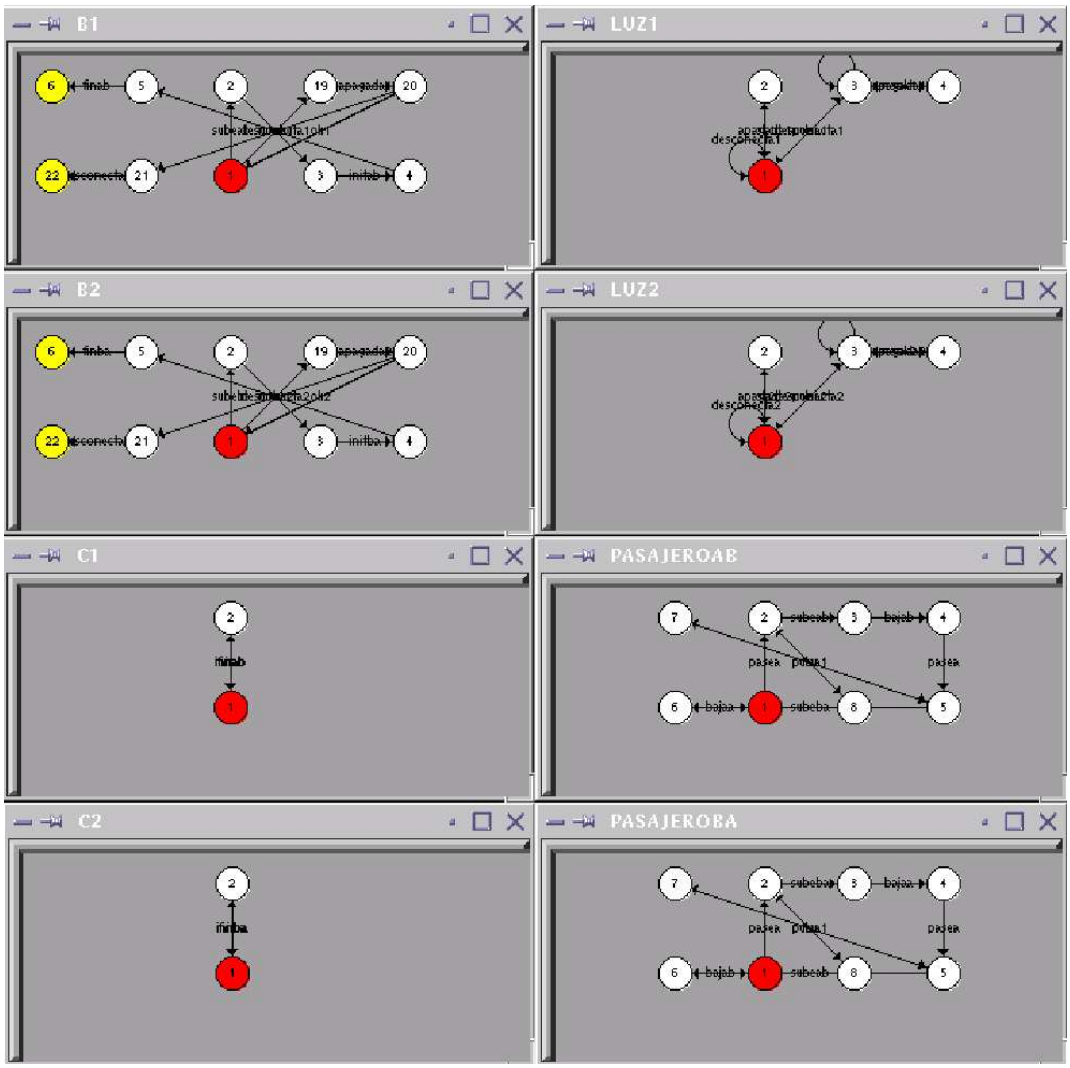


Figura 9.13: Browser de grafos

tras lo cual será requerido para que indique el tiempo que desea que transcurra. Este valor tendrá una cota superior, en función de las acciones urgentes que puedan existir.

En cada instante, todas las transiciones que estén habilitadas para ser ejecutadas aparecerán pintadas de rojo. Para ejecutarlas bastará con pulsar con el ratón sobre ellas. Si esta transición está etiquetada con una acción perteneciente al conjunto de sincronización con otro proceso, la transición se pintará de azul, y el sistema quedará esperando a que se seleccione en la ventana correspondiente al otro proceso la acción con la que se sincroniza, es decir, se deberá seleccionar de forma explícita

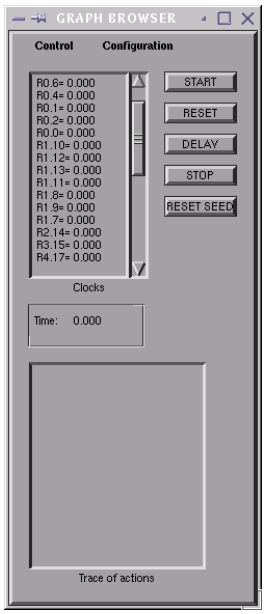


Figura 9.14: Ventana de simulación

la acción en concreto con la que se sincroniza entre las pintadas en rojo.

Al mostrarse los grafos, podemos pulsar sobre cualquier arco o nodo con el botón derecho del ratón para obtener información sobre el mismo (Figura 9.16).

9.3.2. Trazas

Tras realizar una simulación del sistema existe la posibilidad de salvar en un fichero la traza de las acciones ejecutadas hasta un determinado momento. La información almacenada está compuesta por el nombre de la acción ejecutada y el instante de tiempo en el que ha sido ejecutada.

Nuestra intención es que estas trazas salvaguardadas puedan ser posteriormente importadas, para así poder reproducir un experimento. Esta característica de importación de trazas no se encuentra implementada todavía.

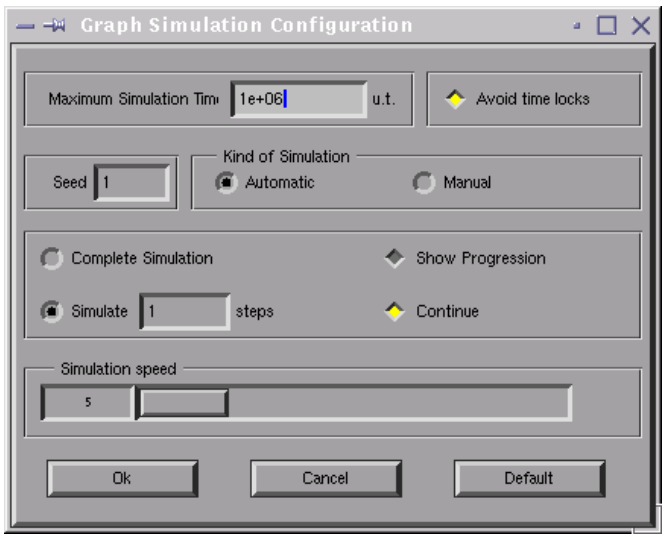


Figura 9.15: Cuadro de configuración de la simulación

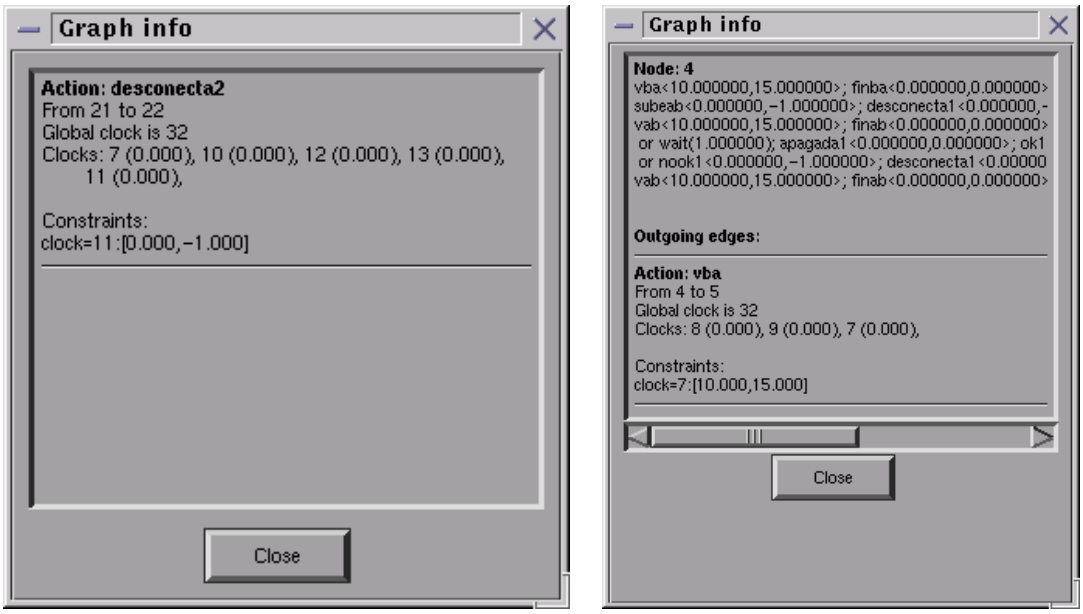
9.4. Redes de Petri con arcos temporizados

Recientemente se ha incorporado a la herramienta la posibilidad de realizar la traducción automática a redes de Petri con arcos temporizados, de acuerdo con la definición formal presentada en el capítulo 6, obteniendo como resultado una única red de Petri con arcos temporizados para todo el sistema. Actualmente, la herramienta solamente permite la visualización de las redes de Petri en modo texto. Esperamos finalizar en breve plazo la componente encargada de la visualización gráfica.

Así mismo, se permite salvar la red de Petri en un fichero, con el fin de poder importarla posteriormente y utilizarla directamente. De hecho, nuestra intención es implementar un nuevo componente en la herramienta para la creación y edición de redes de Petri temporizadas de forma directa.

La herramienta ya implementa un pequeño kernel de redes de Petri con arcos temporizados, que permite incluso su simulación, aunque al no disponer de una interfaz gráfica, esta característica aún no ha podido ser utilizada.

El fichero de texto correspondiente a la red de Petri con arcos temporizados obtenida para el ejemplo de las barcas es el mostrado en la Figura 9.17.



(a) De Arcos (b) De Nodos

Figura 9.16: Información sobre elementos del grafo

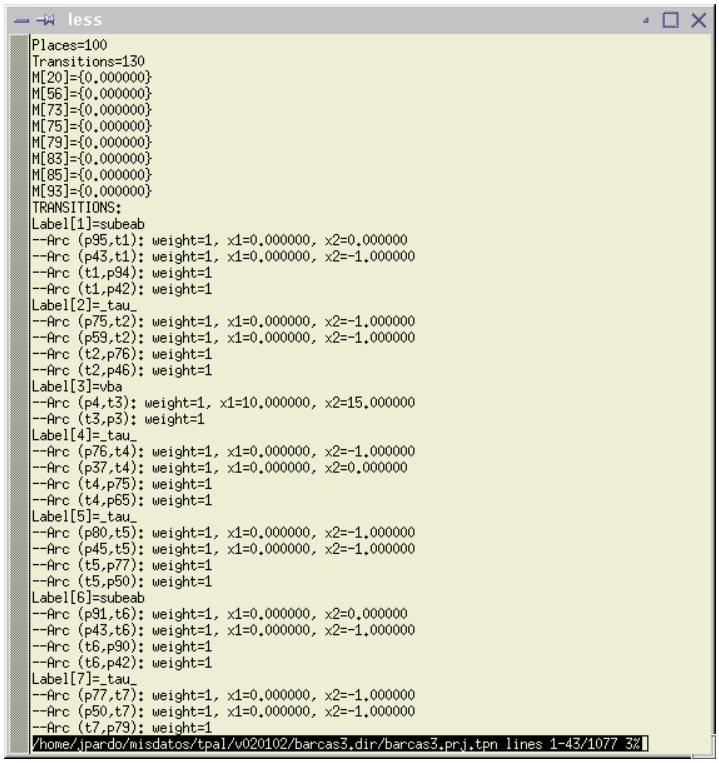


Figura 9.17: Visualización de la red de Petri generada

Capítulo 10

Conclusiones y Trabajos futuros

En esta tesis hemos presentado un álgebra de procesos temporizada denominada TPAL. En ella, además de los operadores básicos de LOTOS, se ha introducido un conjunto de operadores temporales que nos permiten definir restricciones temporales en las especificaciones. Los operadores introducidos han sido un operador de acción prefijo temporizado, un operador de espera (*wait*) y un prefijo de acción urgente.

Posteriormente se ha ampliado este álgebra con algunos operadores probabilísticos dando lugar a un algebra temporizada y probabilística que hemos denominado TPPAL. Estos operadores probabilísticos nos permitirán introducir aspectos probabílisticos en nuestras especificaciones, de modo que podremos especificar sistemas cuyo comportamiento puede ser caracterizado probabilísticamente, como los sistemas tolerantes a fallos.

También se ha presentado un nuevo modelo para la especificación de sistemas concurrentes, como son los *grafos de estados dinámicos*. Este modelo presenta grandes similitudes con los autómatas temporizados, diferenciándose de éstos en la utilización de los relojes. En el modelo de grafos de estados dinámicos los relojes son actualizados al instante de tiempo en que se ejecuta una transición, mientras que en los autómatas temporizados son reseteados. La ventaja es que podemos saber fácilmente el tiempo transcurrido para cada componente secuencial. Por contra, se hace más complicada la verificación del modelo mediante técnicas de model-checking, ya que dichas técnicas se basan en la creación de grafos de regiones.

Junto a la definición del álgebra y del modelo de grafos de estados dinámicos se han definido los mecanismos para la traducción de las especificaciones realizadas

en el álgebra a otros modelos gráficos, concretamente a Redes de Petri con arcos temporizados etiquetadas y a autómatas temporizados. Esta traducción nos permite realizar una sola vez el esfuerzo de especificación de un sistema, usando un modelo en el que la especificación resulta relativamente fácil y a continuación de forma autómatica, obtener otros modelos que facilitan el análisis del sistema.

En el desarrollo de la tesis hemos comenzado por definir la sintaxis de un subconjunto del lenguaje TPPAL, el cual contiene solamente los términos temporizados y al cual hemos denominado TPAL. La sintaxis de este lenguaje mantiene los operadores de LOTOS de elección externa, paralelismo, ocultamiento y acción interna y además se han añadido los operadores de prefijo temporizado y *wait*. Junto a esta sintaxis se ha definido la semántica operacional de TPAL mediante un sistema de transiciones etiquetado.

A continuación se ha presentado el nuevo modelo introducido en esta tesis, los grafos de estados dinámicos, junto con la definición de la traducción del álgebra a este nuevo modelo.

Posteriormente se ha presentado la traducción a autómatas temporizados. Para realizar esta traducción hemos partido de los grafos obtenidos en lugar de hacerlo a partir del álgebra.

Para finalizar el tratamiento del sublenguaje temporizado TPAL hemos procedido a definir la traducción a redes de Petri con arcos temporizados. Esta traducción se ha realizado de forma gradual, de modo que en primer lugar solamente se ha presentado la traducción para los procesos finitos, para los cuales se realiza de forma relativamente fácil. Seguidamente, se ha presentado la traducción para términos recursivos imponiendo diversas restricciones para conseguir que esta traducción sea correcta. Finalmente, viendo que algunas restricciones impuestas podían ser relajadas en algunos casos muy especiales, se ha procedido a establecer algunos casos en los cuales podemos levantar esas restricciones.

Una vez finalizada la presentación de las diferentes traducciones del álgebra temporizada a los modelos elegidos se ha procedido a la presentación del lenguaje algebraico TPPAL completo, introduciendo los nuevos operadores probabilísticos y estableciendo las variaciones pertinentes en el lenguaje temporizado.

La coexistencia de una elección probabilística y una externa nos obligó a decidir y establecer la relación existente entre ambos operadores. Sintácticamente se establece una alternancia entre ambos operadores, mientras que semánticamente se establece

que en primer lugar se resolverán las elecciones probabilísticas, para posteriormente resolver las elecciones externas.

Por último, y dado que uno de los objetivos de esta tesis es el de obtener una herramienta asistida por ordenador que permita el análisis y evaluación de los sistemas modelados, se ha presentado la herramienta que nuestro grupo de investigación está desarrollando, y para la cual los modelos teóricos presentados en esta tesis son la base formal que la sustenta.

10.1. Publicaciones

El trabajo desarrollado para esta tesis ha dado como resultado diversos artículos, los cuales han sido presentados en diversos congresos de carácter nacional e internacional.

Tanto en las *VIII Jornadas de Concurrencia* celebradas en Junio de 2000 en Cuenca [PVCR00] como en el congreso *Software Engineering Applied to Networking & Parallel/Distributed Computing, SNPD'00* celebrado en Reims (Francia), en Mayo de 2000, se presentó la definición del lenguaje temporizado, junto con su semántica operacional y la traducción a grafos de estados dinámicos [PVC00].

En la *Asia-Pacific Software Engineering Conference*, celebrada en Macao en diciembre de 2001, se presentó la extensión del lenguaje con los operadores probabilísticos, así como las principales características de la herramienta que nuestro grupo de investigación está desarrollando ([PVCC01]).

Finalmente, en la *International Conference of Theory and Aplication of Petri Nets* celebrada en Adelaide (Australia) en Junio de 2002 se presentó la traducción de los términos temporizados de *TPAL* a redes de Petri con arcos temporizados [VPC02].

10.2. Trabajo futuro

Nuestro trabajo discurrirá por dos vías, que aunque relacionadas, podrán ser desarrolladas independientemente.

La primera de ella se centrará en la parte del desarrollo teórico-formal donde se

llevarán a cabo tareas como:

- Definición de la traducción del lenguaje TPPAL a un modelo apropiado de redes de Petri con arcos temporizados. Para ello estudiaremos los diferentes modelos de redes de Petri que integren aspectos temporales y probabilísticos, para seleccionar el más adecuado para nuestros propósitos. Una vez seleccionado el modelo adecuado procederemos a la definición de la traducción, centrándonos principalmente en los operadores probabilísticos.

- Definición de algoritmos de verificación de los modelos. Para ello se utilizará la técnica de model-checking. Dentro de esta línea de trabajo, este apartado es el que se considera más interesante y en el que se centrará la mayor parte de nuestro esfuerzo.

La segunda línea de trabajo se centrará en la herramienta TPAL, de modo que sea dotada de unas características y funciones que nos permitan la utilización de la misma para el estudio de casos reales. Para conseguir este objetivo, algunas de las tareas planeadas son:

- Completar el traductor de las especificaciones algebraicas a grafos de estados dinámicos de modo que permita la traducción de especificaciones probabilísticas.

- Incorporación de la posibilidad de realizar el análisis y verificación de los sistemas mediante técnicas de model-checking.

- Implementación de un visualizador de redes de Petri, que permita una visualización estática, y que permita la simulación de las redes generadas. Para esta ejecución será necesario la conexión de dicho visualizador con el núcleo motor de la simulación, el cual se encuentra ya implementado.

- Incorporación a la herramienta de un editor de redes de Petri que permita la creación y edición de redes de Petri temporizadas directamente, de forma gráfica, sin necesidad de realizar previamente la especificación algebraica.

- Implementación de algoritmos de verificación de propiedades mediante las redes de Petri.

- Integración de nuestra herramienta TPAL con otras herramientas ya existentes, de modo que los resultados obtenidos por nuestra herramienta puedan ser tratados mediante esas otras herramientas.

Para finalizar, deseo reseñar que tanto el trabajo realizado para esta tesis como el trabajo planteado como continuación de la misma se desarrollan dentro de los siguientes proyectos de investigación:

- Desarrollo Formal de Sistemas Distribuidos (TIC97-0669-C03-02). Este proyecto formaba parte de un proyecto coordinado llevado a cabo en colaboración con la Universidad Complutense de Madrid y la Universidade de Vigo. Fue desarrollado entre los años 1997 y 2000, con un total de 9 participantes en nuestro subproyecto.
- Evaluación de Rendimientos de Sistemas Distribuidos (TIC2000-0701-C02-02). Este proyecto es un subproyecto de un proyecto más ambicioso titulado "Desarrollo Formal de Sistemas Basados en Agentes Móviles y Evaluación de Rendimientos" el cual se está llevando a cabo en colaboración con la Universidad Complutense de Madrid y cuya duración es hasta finales del año 2003.

Bibliografía

[ACD90] R Alur, C. Courcoubetis, and D.L. Dill. Model-Checking for Real-Time Systems. In *Proceedings of the 5th Annual Symposium on Logic in Computer Science*, pages 414–425. IEEE Computer Society Press, 1990.

[AD90] R. Alur and D.L. Dill. Automata for Modelling Real-Time Systems. In *Proceedings of ICALP'90*, number 443 in Lecture Notes in Computer Science, pages 322–335. Springer-Verlang, 1990.

[AMBB+85] M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, and A. Cumani. On Petri Nets with Stochastic Timing. In *Proc of th International Workshop on Timed Petri Nets*, pages 80–87. IEEE Computer Society Press, 1985.

[AN01] Parosh A. Abdulla and Aletta Nylen. Timed Petri Nets an BQOs. In *Proc 22nd International Conference on Theory and Application of Petri Nets.*, volume 2075 of *Lecture Notes in Computer Science*, pages 53–70, 2001.

[BB87] T. Bolognesi and WE. Brinksma. Introduction to the ISO Specification Language LOTOS. *Computer Networks and ISDN Systems*, 14(1):25–59, 1987.

[BDH92] E. Best, R. Devillers, and J. Hall. The Petri Box Calculus: A New Causal Algebra with Multi-label Communication. *Advances in Petri Nets,1992. Lecture Notes in computer Science*, 609:21–69, 1992.

[BLPY95] J. Bengtsson, K. Larsen, P. Peterson, and Wang Yi. Uppaal.- A Tool Suite for Automatic Verfication of Real-Time Systems. In *Proc of 4th*

Dimacs Workshop on Verification and Control of Hybrid Systems, New Jersey, October 1995.

[BLT90] T. Bolognesi, F. Lucidi, and S Trigila. From Timed Petri Nets to Timed LOTOS. In *Proceedings of Tenth International IFIP WG6.1 Symposium on Protocol Specification Testing and Verification*. North-Holland, 1990.

[Bow96] Fred D. J. Bowden. Modelling time in Petri nets. In *Proc. Second Australia-Japan Workshop on Stochastic Models*, 1996.

[Buc62] R. Buchi. On a Decision Method in Restricted Second-Order Arithmetic. In *Proceedings of the Internationa Congress on Logic,Methodology an Philosophy of Science1960*, pages 1–12. Standford University Press, 1962.

[CdFV96] F. Cuartero, D. de Frutos, and V. Valero. PCSP: A Denotational Model for Probabilistic Processes. In *Proceedings of Third AMAST Workshop on Real-Time Systems*, 1996.

[CdFV97] F. Cuartero, D. de Frutos, and V. Valero. A Sound and Complete Proof System for Probabilistic Processes. In *ARTS'96, LNCS 1231*, 1997.

[Cho74] Y. Choueka. Theories of Automata on ω-tapes: a Simplified Approach. *Joyurnal of Computer and Syustem Sciences*, 8:117–141, 1974.

[CMS99] Antonio Cerone and Andrea Maggiolo-Schettini. Time-Based Expressivity of time Petri Nets for System Specification. *Theorical Computer Science*, 216(1-2):1–53, 1999.

[DB96] P.R. D'Argenio and E. Brinksma. A Calculus for Timed Automata (Extended Abstract). In B. Jonsson and J. Parrow, editors, *Proceedings of the 4th International School and Symposium on Formal Techniques in Real Time and Fault Tolerant Systems,* Uppsala, Sweden, volume 1135 of *LNCS*, pages 110–129. SPRINGER, 1996.

[DIT95] DIT UPM. *LOtos LAboratory. User Manual (Version 3R6)*, February 1995.

[EM85] H Ehrig and B. Mahr. Fundamentals of Algebraic Specification. *Bull. Euro. Association Theoretical Computer Science*, 1985.

[Gol88] U. Goltz. On Representing CCS Programs by Finite Petri Nets. *MFCS Lecture Notes in Computer Science*, 324:339–350, 1988.

[Gro93] Jan Friso Groote. Transition system specifications with negative premises. *Theoretical Computer Science*, 118:263–299, 1993.

[Han93] Hans-Michael Hanisch. Analysis of Place/Transition Nets with Timed-Arcs and its Application to Batch Process Control. In *Application and Theory of Petri Nets*, volume 691 of *Lecture Notes in Computer Science*, pages 282–299, 1993.

[HJ90] Hans Hansson and Bergt Jonsson. A Calculus for Communicating Systems with Time and Probabilities. In *11 IEEE Real-Time Systems Symposium*. IEEE Computer Society Press, 1990.

[HNSY] Thomas A. Henzinger, Xabier Nicollin, Joseph Sifakis, and Sergio. Yovine. Symbolic Model Checking for Real-Time Systems. *Information and Computation*, 3(2):193–244.

[Hoa78] C.A.R. Hoare. Communicating Sequential Processes. *Communications of the ACM*, 21(8):666–677, 1978.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall International, 1985.

[Kat98] Joost-Pieter Katoen. *Concepts, Algorithms, and Tools for Model Checking*. Friedrich-Alexander Universität Erlangen-Nürnberg, 1998. Lecture Notes of the Course "Mechanised Validation of Parallel Systems".

[LOT96a] Juan Quemada Editor, Revised Working Draft on Enhancements to LOTOS (v2) ISO/IEC JTC1/SC21/WG7, January 1996. N10108 Project 1.21.20.2.3. Output document of the Ottawa meeting.

[LOT96b] Juan Quemada editor, Revised Working Draft on Enhancements to LOTOS (v4) ISO/IEC JTC1/SC21/WG7, September 1996. N1173 Project 1.21.20.2.3. Output document of the Kansas City meeting.

[LOT98] Final Commite Draft on Enhancements to LOTOS ISO/IEC JCT1/SC21/WG7, May 1998. Project 1.21.20.2.3.

[LPY95] Kim G. Larsen, Paul Petterson, and Wang Yi. Compositional and Symbolic Model-Checker of Real-Time Systems. In *Proceedings of the 16th Real-Time Systems Symposium*, pages 76–87. IEEE Computer Society Press, 1995.

[LS89] K.G. Larsen and A. Skou. Bisimulation through Probabilistic Testing. In *Proceedings of 16th ACM Symp. on Principles of Programming Languages*, Austin, TX, 1989.

[McN66] R. McNaughton. Testing and Generating Infinite Sequences by a Finite Automaton. *Information and Control*, 9:521–530, 1966.

[Mer74] P. Merlin. *A Study of the Recoverability of Communication Protocols.* PhD thesis, Computer Science Dep. Univ. California, 1974.

[Mil80] Robin Milner. *A Calculus of Communicating Systems.* Lecture Notes in Computer Science 92. Springer-Verlang, 1980.

[Mil89] Robin Milner. *Communication and Concurrency.* Prentice-Hall International, 1989.

[MT90] F. Moller and C. Tofts. A temporal Calculus of Communicating Systems. In *Proc. CONCUR 90*, number 458 in Lecture Notes in Computer Science, pages 401–415, Amsterdan, 1990.

[NS90] Xabier Nicollin and J. Sifakis. The Algebra of Timed Processes ATP: Theory and Application. *Information and Computation*, December 1990.

[NSY92] Xabier Nicollin, Joseph Sifakis, and Sergio Yovine. Compiling Real-Time Specification into Extended Automata. *IEEE TRansactions onm Software Engineering*, 18(9):794–804, 1992.

[NSY93] Xabier Nicollin, Joseph Sifakis, and Sergio Yovine. From ATP to Timed Graphs and Hybrid Systems. *Acta informatica*, (30):181–202, 1993.

[Ort90] Yolanda Ortega. *En busca del Tiempo Perdido.* PhD thesis, Dpto. Informática y Automática (UCM), 1990.

[Pet62] C.A. Petri. *Kommunikation mit Automaten Ph.D dissertation.* PhD thesis, University of Bonn, Bonn, West Germany, 1962.

[Plo81] G.D. Plotkin. A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, Computer Science Department, Aarhus Universtiy, 1981.

[PVC00] J.J. Pardo, V. Valero, and F. Cuartero. A dynamic state graph for a timed process algebra. In *Proc. of SNPD'00*, pages 199–209, May 2000.

[PVCC01] J. Pardo, V Valero, F Cuartero, and D. Cazorla. Automatic translation of tpal specification into dynamic state graphs. In *Proc of APSEC'01*. IEEE Computer Society Press, 2001.

[PVCR00] Juan J. Pardo, Valentn Valero, Fernando Cuartero, and M.Carmen Ruiz. A dynamic state graph for a timed process algebra. In *Actas VIII Jornadas de Concurrencia*, pages 35–54. Servicios Publicaciones UCLM, June 2000.

[QAdF89] J. Quemada, A. Azcorra, and D. de Frutos. A timed calculus for lotos2. *Proc. FORTE 89*, 1989.

[Ram74] C. Ramchandani. Analysis of Asyncronous Concurrent Systems by Timed Petri Nets. Technical Report 120, Project MAC, 1974.

[RR88] G.M. Ree and A.W. Roscoe. A Timed Models for Communicating Sequential Processes. *Theoretical Computer Science 58*, (1988):249–261, 1988.

[SB] Richard H. Sloan and Ugo. Buy. Stubborn Sets for Real-Time Petri Nets. *Formal Methods in System Design*, 11(1):23–40.

[Sif77] J. Sifakis. Use of Petri Nets for Performance Evaluation. In *Proc of the Third International Symposium IFIP W.G.7.3 Measuring, Modelling and Evaluating Computer Systems*, pages 75–93. Elsevier Science Publishers, 1977.

[Tau89] Dirk Taubner. Finite Representation of CCS and TCSP Programs by Automata and Petri Nets. *Lecture Notes in Computer Science*, 369, 1989.

[Var87] M. Vardi. Verification of Concurrent Programs- the Automata-Theoretic Framework. In *Proceedings of the Second IEEE Symposium on Logic in Computer Science*, pages 167–176, 1987.

[vdA93] W.M.P. van der Aalst. Interval Timed Coloured Petri Nets and their Analysis. *Lecture Notes in computer Science*, 691:451–472, 1993.

[vdAO95] W.M.P. van der Aalst and M.A. Odijk. Analysis of Railway Station by Means of Interval Timed Coloured Petri Nets. *Real-Time Systems*, 9:241–263, 1995.

[VdFC99] V. Valero, D. de Frutos, and F Cuartero. Decidability of the Strict Reachability for Timed-Arc Petri Nets. In *Proc 8th Int. Workshop on Petri Nets and Performance Models PNPM'99*, pages 188–196, 1999.

[vGSS95] R. van Glabbeek, S.A. Smolka, and B.U. Steffen. Reactive, Generative and Stratified Models of Probabilistic Processes. *Information and Computation*, 121(1):59–80, 1995.

[vGSST90] R. van Glabbeek, S.A. Smolka, B.U. Steffen, and C.M.N. Tofts. Reactive, Generative and Stratified Models of Probabilistic Processes. In *Proceedings of 5th Annual IEEE Symposium on Logic in Computer Science*, pages 130–141, Philadelphia, PA, 1990.

[VPC02] Valentín Valero, Juan J. Pardo, and Fernando Cuartero. Translating TPAL Specifications Into Timed-Arc Petri Nets. In *Proccedings of ICTAPN'02*, 2002. Pendiente de Publicacion.

[Wal83] B. Walter. Timed Petri-Nets for Modelling and Analyzing Protocols with Real-Time Characteristics . In *Proc 3rd IFIP Workshop on Protocol Specification , Testing and Verification*, Noth-Holland, 1983.

[YS] T. Yoneda and B. Schlingloff. Efficient Verification of Parallel Real-Time Systems. *Formal Methods in System Design*, 11(2):197–215.